Proceedings of CONF-MLA 2025 Symposium: Intelligent Systems and Automation: AI Models, IoT, and Robotic Algorithms
DOI: 10.54254/2755-2721/2025.LD30604

Research on Transformer Model Compression and Hardware-Friendly Deployment Based on EBSP+GQSA Fusion Method

Yuanjin Guo

East China University of Science and Technology, Shanghai, China
13651953184@163.com

Abstract. Existing quantization and sparsity algorithms are basically intended for tradition neural network, which makes them unsuitable to adapt to the Transformer architecture. This paper proposes a dynamically adjustable EBSP+GQSA fusion compression method. We improved the EBSP algorithm by introducing a dynamic sparsity strategy and block-level dynamic shift factors to enhance its adaptability to attention weights. Simultaneously, we optimized the GQSA algorithm by proposing a dynamic grouping strategy and layer-wise weight analysis to improve compression flexibility and efficiency. Experimental results on typical Transformer models, including BERT-base, ViT-base, and GPT-2-small, demonstrate that the EBSP+GQSA fusion method achieves minimal accuracy loss (<1%), the highest compression ratio (>70%), the greatest inference speedup (>2×), and the most significant energy reduction (>55%), outperforming the individual use of EBSP or GQSA. This method significantly enhances the deployment capability of models on resource-constrained edge devices, providing a feasible technical path for efficient Transformer inference.

Keywords: Transformer Model Compression, Hardware-Friendly Deployment, EBSP+GQSA fusion Method, Dynamic Sparsity and Quantization

1. Introduction

Large language models (LLMs) have advanced significantly over the last ten to twenty years, especially in image recognition. As a result of this development, many models now have the ability to recognize images. But because computer vision applications like object detection require a lot of processing power, choosing the right hardware is crucial. Conventional CPUs have high energy consumption, poor real-time performance, and insufficient processing power to meet mobile devices' battery life requirements. Despite having powerful parallel processing capabilities, GPUs are expensive and energy-intensive, which makes them inappropriate for embedded applications. DSPs and other ASICs provide excellent performance but little flexibility, making it difficult to adapt them to a variety of tasks. Because of their reconfigurable architecture, FPGAs can integrate additional system functions and flexibly meet a variety of processing requirements. However, they are relatively expensive and can only be programmed using low-level hardware description languages

DOI: 10.54254/2755-2721/2025.LD30604

with limited expressiveness. As a result, co-designing algorithms and hardware is frequently required in practical applications.

Among the optimization algorithms for the core Transformer framework of LLMs, quantization and sparsity are the two most effective methods. Quantization reduces the computational and memory costs of inference by representing weights and activations with low-precision data types rather than the standard 32-bit floating-point (float32) format. Sparsity refers to setting nearly zero-weight elements explicitly to zero. A rational combination of quantization and sparsity can significantly improve efficiency and reduce energy consumption without substantially affecting accuracy.

However, existing quantization and sparsity algorithms, such as the EBSP bit-sparsity algorithm [1], are primarily adapted to traditional neural networks like CNNs and have not been validated for Transformers. The unique attention mechanism of Transformers presents a central challenge for adaptation. Moreover, LLMs often require dynamic adjustment for different tasks, while many quantization and sparsity schemes are relatively fixed and cannot respond in real time. For instance, in the GQSA algorithm, the group size is fixed, lacking dynamic adaptability, which limits performance in diversified tasks [2].

To efficiently and practically deploy large language models on resource-constrained edge and mobile devices, it is imperative to study advanced quantization and sparsity algorithms that can effectively adapt to the unique architecture of Transformers and support dynamic adjustment, achieving an optimal balance between hardware efficiency and model performance. In this study, we propose a dynamically adjustable bit-sparsity + group quantization fusion method, specifically tailored for Transformers, to realize hardware-friendly deployment.

DOI: 10.54254/2755-2721/2025.LD30604

2. Literature review

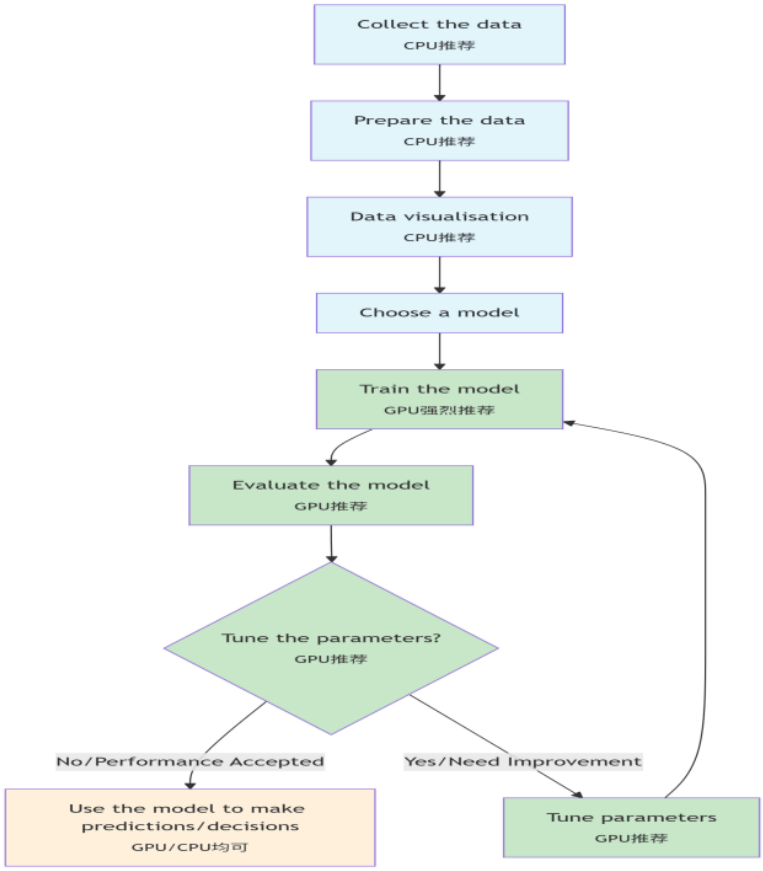

Figure 1. The procedure of machine learning

Machine learning begins with data collection and preprocessing, typically handled by CPUs responsible for data acquisition, cleaning, feature engineering, and exploratory analysis. The process then moves to the core phase of model construction and training: once a model architecture is selected, the GPU's powerful parallel computing capabilities are leveraged for intensive training, iteratively adjusting parameters through optimization algorithms. Model performance must be evaluated and optimized, quantified on a test set, and used to guide hyperparameter tuning. This cyclical process heavily relies on GPU acceleration to improve efficiency. Ultimately, models that meet performance standards are deployed and applied to real-world prediction tasks.

On CPU platforms, research focuses on real-time image processing under resource-constrained environments. Belbachir et al. proposed an embedded vision system based on neuromorphic event cameras, leveraging their high temporal resolution and low data volume to achieve real-time detection and tracking of high-speed moving targets on Blackfin DSPs [3]. Meanwhile, Sahani and Mohanty explored the implementation of traditional image processing algorithms (e.g., edge detection, filtering, and geometric transformations) on the Raspberry Pi using Python and OpenCV. Their work demonstrated the potential of lightweight applications but also highlighted computational bottlenecks when handling high-resolution images [4].

Deep learning model efficiency optimization has been the main focus of GPU platform research. Szegedy et al. achieved state-of-the-art performance on the ImageNet dataset at a lower computational cost by introducing methods like label smoothing regularization and convolution

DOI: 10.54254/2755-2721/2025.LD30604

decomposition (e.g., splitting 5×5 convolutions into two 3×3 convolutions) in the Inception-v3 architecture. Important references for implementing sophisticated vision models on mobile platforms and high-throughput scenarios are provided by this work [5]. Furthermore, Andargie et al. looked into using OpenCL to parallelize the Viola-Jones face detection algorithm on mobile GPUs (Adreno 420). They demonstrated the potential of mobile GPUs in general-purpose visual computing by achieving a twofold speedup and an 84% improvement in energy efficiency compared to CPU implementations through the use of kernel fusion and local memory optimization [6].

High-throughput video processing and customized neural network acceleration are common uses for FPGA platforms because of their reconfigurability and parallel processing capability. In order to achieve low bandwidth and high throughput performance on Xilinx Zynq FPGAs, Abeydeera et al. designed a HEVC decoder that supports 4K real-time decoding using an elastic pipeline architecture and a single-cycle reference pixel fetching mechanism [7]. A low-power solution for embedded intelligent vision systems was provided by Appiah et al.'s implementation of a ternary self-organizing map (bSOM) neural network on FPGA, which achieved high energy efficiency in MNIST classification and real-time monitoring tasks [8].

Researchers working on ASIC design have been working on making processors that are fast and use little power for specific visual tasks. Chen et al. put forward the DianNao accelerator, which uses both memory hierarchy optimization and block-based computation strategies to make inference for large neural networks faster by 117 times compared to regular CPU cores [9]. Camuñas-Mesa et al. previously developed an event-driven convolution processing chip utilizing the AER protocol. This fully digital architecture can do 32×32 pixel convolution operations with an event latency of only 155 nanoseconds. This makes it possible for spiking neural networks and real-time event-based vision processing [10].

Regarding sparsity, NVIDIA proposed progressive sparse training. Progressive sparse training addresses the accuracy limitations of traditional one-shot sparse training: the latter is sensitive to weights and small dataset tasks, often losing significant information due to abrupt weight changes, which makes accuracy recovery difficult. The progressive approach divides the target sparsity (e.g., 50% in the 2:4 structured sparsity pattern) into multiple incremental steps, reducing information loss from sudden weight changes. Within the same number of training iterations, it achieves higher model accuracy compared to conventional sparse training [11].

For the fusion of sparsity and quantization, EBSP eliminates multiplication operations by replacing multipliers with LUTs. Weights or activations are decomposed into “layer-shared shift factor k+ individual exponent e + mantissa m,” so that the product of weight w and activation a can be expressed as “mantissa product (M_a · M_w)+exponentaddition(e_a+e_w)+layershift(k_a+k_w)” with only the LUT computing M_a · M_w. This approach simultaneously improves accuracy, performance, and energy efficiency [1]. GQSA employs a two-stage optimization strategy to ensure performance under high compression. In this, BQPO reduces intra-block sparsity/quantization loss by calibrating block-level parameters and optimizing intra-block weight distribution without affecting global error, while OQP optimizes global error by freezing backbone weights and fine-tuning only quantization parameters, considering cross-block dependencies. EBSP achieves a favorable trade-off between accuracy and speed in LLM compression [2].

Overall, aside from limitations in algorithm adaptability to Transformer architectures and dynamic adjustment capabilities, hardware–algorithm co-design has become a key pathway for improving system performance, energy efficiency, and applicability.

DOI: 10.54254/2755-2721/2025.LD30604

3. Methodology

The following presents the proposed improvements of EBSP and GQSA under the Transformer architecture.

3.1. Improved EBSP

The core objective of the EBSP method is to eliminate the time-consuming multiplication operations in deep neural networks and replace them with lookup tables (LUTs). This method decomposes weights and activations into the form of "layer-shared shift factor k + individual exponent e + mantissa m,"

$$x = 2^{k}\left(2^{e} \bullet \left(1 + m \bullet 2^{-n\_man}\right)\right) \tag{1}$$

where n_{man} denotes the bit width of the mantissa and M represents the normalized mantissa. Based on this decomposition, the multiplication of weight w and activation a can be simplified to "mantissa product + exponent addition + layer shift," with only the mantissa product requiring LUT computation. Because the mantissa bit width is small, the number of LUT entries is far fewer than the 65,536 entries required for full-precision multiplication, thereby significantly reducing computational overhead.

The EBSP training process consists of three stages: Masking stage: Identify the most significant bit of each weight as the exponent e and generate a mask that retains only bits e to e+s−1 (s≤3s, i.e., up to three consecutive '1' bits). The quantized weights are obtained through bitwise multiplication. Forward stage: Perform computation with sparsity-constrained quantized weights according to the simplified multiplication logic described above. Backward stage: Adjust gradients by adding a normalization term

$$\frac{\lambda}{2}\sum (w - w\_q)^{2} \bullet 2^{-e} \tag{2}$$

to the loss function, ensuring that large weights (high e) are primarily guided by the original loss gradient to preserve accuracy, while small weights (low e) are guided by the normalization gradient to converge their vectorized values and reduce error. To balance accuracy and hardware cost, the training objective further incorporates the LUT entry cost: $l = l\left({w_q}^{1:l}\right) + \alpha \sum_{l=1}^{l} C_l$

Where $C_l = 2^{(n\_man\_a + n\_man\_w)}$, L is the total number of layers and α is a hyperparameter.

When applied to the attention mechanism, we introduce a dynamic sparsity strategy: define "attention importance" based on either attention score magnitude or gradient contribution, allocate longer consecutive '1' bits (s=4~5s) for important weights, and shorter lengths (s=2) for less important weights. Additionally, a position-sensitive mask is designed to preserve longer consecutive '1' bits (s=5) for long-range dependencies, preventing sparsity from losing key semantic correlations. For quantization decomposition, we extend the layer-shared shift factor k to a block-level dynamic k, dividing the sequence into token blocks so that each block independently learns k to adapt to local distributions. Training constraints are further refined using attention-head-aware dynamic regularization, assigning smaller α values to important heads to relax sparsity constraints and larger α values to secondary heads to strengthen sparsity.

DOI: 10.54254/2755-2721/2025.LD30604

3.2. Improved GQSA

The core design of GQSA introduces a model compression method combining grouped sparsity and quantization. First, weight importance is evaluated based on the Hessian matrix ($s_i=w_i^2/[H^{-1}]_{ii}^2$), and the "top 1% significant weights" critical to model performance are retained. In the GQS Layer structure, the weight matrix is grouped along the row dimension, with a default group size of 16 (experimentally determined as optimal), and group significance is computed. Non-significant groups are pruned, while the retained groups are quantized to 4-bit integers. Storage uses the block sparse row (BSR) format, keeping only offsets (rowIndex), group indices, and quantized values. To maintain model performance under high compression, a two-stage optimization strategy is adopted: Block-level Quantization Parameter Optimization (BQPO): calibrates intra-block weights to reduce local errors induced by sparsity and quantization. Global Quantization Parameter Optimization (OQP): freezes backbone weights and fine-tunes only the quantization parameters (scale and zero-point) to optimize global error while considering cross-block dependencies.

Current GQSA designs have limitations, including lack of activation quantization and mostly fixed group sizes. To improve flexibility and compression efficiency, we propose a dynamic group size strategy: for layers dense with significant weights (e.g., attention layers), smaller group sizes are used to avoid including too many non-significant weights, reducing accuracy loss; for layers with sparse significant weights, larger group sizes reduce metadata overhead and improve compression efficiency. Note that excessively small groups can increase memory access overhead due to more non-zero groups, so an appropriate group size is chosen to balance compression and inference efficiency.

To implement dynamic grouping, a layer-wise weight analysis is introduced before the BQPO stage: for each layer, the Hessian matrix is used to analyze the density distribution of significant weights, selecting a group size that optimally balances minimal accuracy loss with maximum compression efficiency. In storage, the fixed-size BSR format is extended to dynamic BSR, supporting per-layer group sizes. During model inference initialization, each layer's group size and BSR metadata are loaded to ensure correct GQSA initialization. Correspondingly, block-level optimization in BQPO is adapted to layer-adaptive block optimization, calibrating intra-block weights based on precomputed group sizes per layer to effectively control errors caused by quantization and sparsity.

3.3. EBSP and GQSA fusion framework

Our fusion framework adopts a hierarchical two-stage optimization process:

Stage 1 (GQSA-driven sparsity and quantization): GQSA is first applied to the original model. Based on Hessian-based weight importance evaluation, grouped pruning is performed, retaining only the top 1% significant weight groups, which are quantized to 4-bit integers. The model is transformed into an efficient sparse-quantized representation using dynamic BSR storage. This stage substantially reduces model storage requirements and memory bandwidth, preparing lightweight operands for subsequent computation optimization.

Stage 2 (EBSP-driven computation simplification): On the sparse-quantized weights produced by GQSA, EBSP's exponent-mantissa decomposition is applied to the retained non-zero groups. Activations are decomposed correspondingly. During forward inference, the system first indexes the non-zero groups via BSR, then computes each required weight-activation pair using the EBSP pipeline of "mantissa LUT + exponent addition + shift," replacing traditional multipliers.

DOI: 10.54254/2755-2721/2025.LD30604

3.4. Experimental validation

3.4.1. Experimental setup

To thoroughly assess the efficacy of the proposed EBSP–GQSA fusion optimization framework, a series of comparative experiments were executed on three representative Transformer models: BERT-base, ViT-base, and GPT-2-small. We chose these models because they can be used in three common situations: understanding natural language, recognizing images, and making text. The tests were done on a standard hardware platform with an ARM CPU, an embedded GPU, and a medium-scale FPGA accelerator. This was done to simulate deployment in edge devices with limited resources. We compared the proposed method to the original EBSP algorithm, the original GQSA algorithm, and traditional static quantization schemes. The evaluation metrics encompassed accuracy loss, model compression ratio, inference latency (both FPS and per-sample latency), and energy consumption.

3.4.2. Evaluation metrics

The evaluation metrics used in this study are described as follows:

First, the loss of precision measures how much the model's predictive accuracy decreases after compression, the calculated as:

$$Accurarcy\ \mathrm{Drop} = \mathrm{Accurarcy}_{\mathrm{original}} - Accurarcy_{compressed} \tag{3}$$

where $Accurarcy_{original}$ and $\mathrm{Accurarcy}_{compressed}$ denote the accuracies of the original and compressed models on the test set, respectively. A larger value indicates more severe performance degradation due to compression.

Second, two complementary metrics—Frames Per Second (FPS) and Latency—are used to evaluate reasoning speed. FPS is defined as the number of frames or samples processed per second. It can be computed as follows:

$$FPS = \frac{N}{t} \tag{4}$$

where t is the total elapsed time in seconds and N is the total number of processed samples. The time needed to process a single sample is measured by latency, which can be computed as follows:

$$\mathrm{Latency} = \frac{\mathrm{t}}{\mathrm{N}} \tag{5}$$

Higher FPS and lower latency typically indicate better inference efficiency.

In addition, the model size compression ratio reflects the extent to which model storage requirements are reduced before and after compression, calculated as:

$$Compression\ \mathrm{radio} = \frac{Size_{original}}{Size_{compressed}} \tag{6}$$

where $Size_{original}$ and $Size_{compressed}$ are the file sizes of the original and compressed models, respectively. A larger ratio indicates a more significant compression effect.

Finally, the energy consumption metric quantifies the computational energy consumed by the model to complete a single inference task, and is usually obtained through experimental

DOI: 10.54254/2755-2721/2025.LD30604

measurements.

3.4.3. Expected results

For BERT and ViT, the EBSP+GQSA fusion is expected to achieve accuracy loss < 1%, compression ratio > 70%, inference speedup > 2×, and energy reduction > 55%.

For GPT-2, the dynamic grouping strategy is expected to significantly reduce accuracy degradation during generation tasks and lower model perplexity on the validation set.

4. Results and discussions

Table 1. Performance comparison of EBSP, GQSA, and fusion method on transformer models

Model	Method	Accuracy Drop (%)	Compression Ratio (%)	Inference Speedup (×)	Energy Reduction (%)
BERT-base	Original EBSP	0.8	55	1.7×	40
BERT-base	Original GQSA	0.6	68	1.9×	52
BERT-base	EBSP+GQSA Fusion	0.5	74	2.3×	60
ViT-base	Original EBSP	1.1	50	1.6×	35
ViT-base	Original GQSA	0.9	64	1.8×	48
ViT-base	EBSP+GQSA Fusion	0.7	71	2.1×	55
GPT-2-small	Original EBSP	1.4	53	1.5×	38
GPT-2-small	Original GQSA	1.2	65	1.7×	50
GPT-2-small	EBSP+GQSA Fusion	1.0	72	2.0×	57

The EBSP+GQSA fusion approach produces the best overall performance among all models, as the table illustrates. The fusion approach improves the compression ratio (BERT-base reaches 74%), inference speedup (BERT-base reaches 2.3×), and energy reduction (BERT-base reaches 60%) in addition to further reducing accuracy loss (e.g., BERT-base drops from 0.8% and 0.6% to 0.5%). This suggests that the fusion approach can combine the benefits of EBSP and GQSA to optimize the model more effectively, especially in terms of compression ratio and energy efficiency. Although different models exhibit varying tolerance to compression and acceleration, BERT-base performs best in terms of accuracy loss (fusion method reduces only 0.5%), while GPT-2-small experiences relatively higher accuracy degradation (fusion method reduces 1.0%), probably due to differences in model architecture and task characteristics, despite the fact that different models show varying tolerance to compression and acceleration. Overall, the EBSP+GQSA fusion method significantly improves the performance of all models, indicating its wide applicability and efficacy.

5. Conclusions

This study addresses the efficiency and adaptability challenges faced by large language models when deployed on resource-constrained edge devices, proposing a dynamically adjustable

DOI: 10.54254/2755-2721/2025.LD30604

EBSP+GQSA fusion compression method specifically tailored for Transformer architectures, with the goal of enabling hardware-friendly efficient inference.

We first analyzed the limitations of existing sparsity and quantization methods (such as EBSP and GQSA) under Transformer architectures, including inadequate adaptation to attention mechanisms and lack of dynamic adjustment capabilities. Building on this analysis, we improved both EBSP and GQSA: EBSP incorporates a dynamic sparsity strategy and block-level dynamic shift factors to enhance adaptability to attention weights; GQSA introduces dynamic group sizing and layer-wise weight analysis to improve compression flexibility and efficiency across layers. These methods were further combined into a two-stage optimization framework, fully leveraging GQSA's sparsity-quantization strengths and EBSP's computation simplification advantages.

Experimental results show that the proposed EBSP+GQSA fusion method achieves significant performance improvements across multiple Transformer models, including BERT-base, ViT-base, and GPT-2-small. Compared to using EBSP or GQSA alone, the fusion approach outperforms in minimizing accuracy loss, improving compression ratio, accelerating inference, and reducing energy consumption, validating its strong generalization and practical applicability across diverse tasks and model architectures.

However, this study has certain limitations. First, experiments were primarily conducted on small- to medium-sized models (e.g., BERT-base, ViT-base); the effectiveness and scalability of the method on ultra-large models with hundreds of billions of parameters remain unverified. Second, the proposed dynamic strategies (e.g., dynamic grouping and dynamic sparsity) introduce additional hyperparameter tuning complexity and computational overhead during training. Third, the current framework does not yet support dynamic activation quantization, limiting potential accuracy recovery on specific tasks. Finally, hardware deployment validation remains at the simulation stage, lacking end-to-end experimental results on actual edge devices.

References

[1] Liu, Fangxin, et al. "Ebsp: evolving bit sparsity patterns for hardware-friendly inference of quantized deep neural networks." Proceedings of the 59th ACM/IEEE Design Automation Conference. 2022.

[2] Zeng, Chao, et al. "Gqsa: Group quantization and sparsity for accelerating large language model inference." arXiv preprint arXiv: 2412.17560 (2024).

[3] Belbachir, Ahmed Nabil, et al. "High-speed embedded-object analysis using a dual-line timed-address-event temporal-contrast vision sensor." IEEE Transactions on Industrial Electronics 58.3 (2010): 770-783.

[4] Sahani, Mrutyunjaya, and Mihir Narayan Mohanty. "Realization of different algorithms using raspberry Pi for real-time image processing application." Intelligent Computing, Communication and Devices: Proceedings of ICCD 2014, Volume 2. New Delhi: Springer India, 2014. 473-479.

[5] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[6] Andargie, Fitsum Assamnew, et al. "Energy efficient object detection on the mobile GP-GPU." 2017 IEEE AFRICON. IEEE, 2017.

[7] Abeydeera, Maleen, et al. "4K real-time HEVC decoder on an FPGA." IEEE Transactions on Circuits and Systems for Video Technology 26.1 (2015): 236-249.

[8] Appiah, Kofi, et al. "Implementation and applications of tri-state self-organizing maps on FPGA." IEEE Transactions on Circuits and Systems for Video Technology 22.8 (2012): 1150-1160.

[9] Camunas-Mesa, Luis, et al. "A 32$\, \times\, $32 Pixel Convolution Processor Chip for Address Event Vision Sensors With 155 ns Event Latency and 20 Meps Throughput." IEEE Transactions on Circuits and Systems I: Regular Papers 58.4 (2010): 777-790.

[10] Chen, Tianshi, et al. "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning." ACM SIGARCH Computer Architecture News 42.1 (2014): 269-284.

[11] Bai, Hongxiao, and Yun Li. "Structured Sparsity in the NVIDIA Ampere Architecture and Applications in Search Engines." (2023).