Partially-Structured Transformer Pruning with Patch-Limited XOR-Gate Compression for Stall-Free Sparse-Model Access

Younghoon Byun
POSTECH
Pohang, Republic of Korea
byh1321@postech.ac.kr

Youngjoo Lee
POSTECH
Pohang, Republic of Korea
youngjoo.lee@postech.ac.kr

ABSTRACT

The pruning-based model compression is regarded as an essential technique to deploy the recent large-size transformer models in practical services; however, accessing sparse transformer models cannot reach the ideal speed at all due to the frequent memory stalls for the irregular memory-accessing patterns. Based on the recent XOR-gate compression relaxing the amount of irregular accesses, this work presents a novel partially-structured transformer pruning method dedicated to the interface-friendly compression format. The stall-free memory access is firstly derived by limiting the number of patches per weight, introducing a new trade-off between model quality and effective memory bandwidth. Then, the partially-structured pruning patterns are deployed to provide better accuracy-bandwidth trade-off by significantly reducing the number of correction patches. Adjusting the patch distribution per weight in an aggressive way, the number of limited patches can be even smaller than that of weight bits, further increasing the effective bandwidth for achieving the similar model accuracy. We demonstrate the proposed stall-free XOR-gate compression schemes at pruned DeiT/BERT models on ImageNet/SQuAD datasets, presenting the highest effective bandwidth for accessing sparse transformers compared to the existing stall-based solutions.

KEYWORDS

Deep learning, Model compression, Memory interface, XOR-gate compression

ACM Reference Format:
Younghoon Byun and Youngjoo Lee. 2024. Partially-Structured Transformer Pruning with Patch-Limited XOR-Gate Compression for Stall-Free Sparse-Model Access. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, Moscone west, San Francisco, USA, 6 pages. https://doi.org/10.1145/3649329.3655677

1 INTRODUCTION

In recent years, the development of large-sized transformer models has significantly advanced the state-of-the-art in various natural language processing (NLP) and computer vision tasks [16, 17]. However, deploying these models in practical services often encounters challenges related to their immense computational and memory requirements. To address these issues, model compression techniques have become essential to make transformer models more practical for real-world applications [4]. Among various compression approaches, the model pruning method has been continuously developed to make sparse models by removing less important weights [5, 6]. In general, the sparse model can be compressed by using a specific data format, relaxing the storage overheads and data transfer costs. However, decompressing the compressed model for the following massive-parallel computing units is highly associated with numerous memory stalls caused by irregular data-accessing patterns, making a whole system severely memory-bounded [9]. Due to the randomly-pruned patterns not degrading the model quality, accessing CSR-formatted data naturally suffers from irregular weight accesses [1]; and therefore, it is impossible to guarantee the deterministic decompressing latency from the practical fixed memory bandwidth [7]. The recent XOR-gate compression encodes a sparse matrix into regular weights and irregular error-correction terms called patches, eventually increasing the model-accessing speed than the CSR approach by reducing the amount of irregular memory accesses even for the fixed physical bandwidth [7, 14]. Due to the irregular patches per encoded vector, however, the XOR-gate compression still requires considerable memory stalls with the practical on-chip buffers for synchronizing the encoding vector and the correction information [2]. In general, memory stalls caused by irregular data accesses reduce bandwidth utilization, making low-speed model access far from the ideal bound.

This paper presents advanced model compression methods for allowing stall-free sparse-model access to greatly increase the data transfer speed close to the ideal value. We first propose the patch-limited XOR-gate compression to guarantee the stall-free property for the first time, introducing a new trade-off between model quality and effective memory bandwidth. To minimize the performance drop by limiting the correctable patches per encoded bit, the compression-aware partially-structured transformer pruning is developed according to the pre-defined pruning ratio, achieving better accuracy-bandwidth trade-off by significantly reducing the number of errors in the XOR-gate compression. In addition, we introduce the concept of patch distribution per weight bit and allow the unbalanced distribution to aggressively eliminate redundant dummy patches without degrading the model quality. From the proposed optimization schemes, we can utilize extremely few patches, even smaller than the number of unpruned weights, further increasing the model-accessing speed remarkably. Experimental results show that the proposed approaches successfully increase the model-accessing speed by 18% compared to the recent patch-skipping XOR-gate compression [10].

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06
https://doi.org/10.1145/3649329.3655677

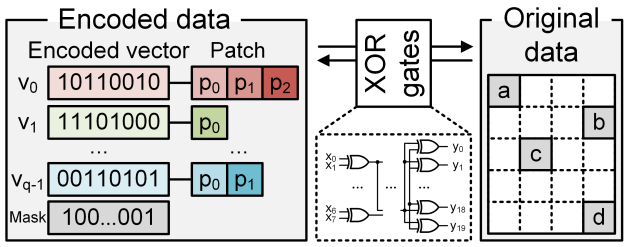

Figure 1: XOR-gate Compression Example.

2 ACCESSING PRUNED TRANSFORMERS

2.1 Model Pruning and Data Compression

Compared with the unpruned transformer with dense weight matrices, the pruned model is associated with sparse matrices by changing less important weights to zero values [11]. In order to preserve the model quality after pruning as much as possible, there have been numerous approaches to evaluate the importance score of each weight, even for the emerging large-size transformers [8]. For example, the recent movement-based pruning identifies redundant weights by capturing the number of moving gradients during the training phase, reporting the perplexity drop of 0.3% by eliminating 80% of total weights ($p = 0.8$) when applied to the GPT-2 model on Wikitext-2 dataset [19]. In addition to the aggressive quantization approaches [4], hence, pruning transformer models is now regarded as an essential optimization method for reducing the overall model size with acceptable performance degradation.

To effectively deploy the pruning results on the practical services, it is necessary to select the proper data compression format dedicated to the sparse matrices for relaxing the storage overheads [2]. However, the on-the-fly decompression step should be realized in advance to activate the processing engines to prepare proper weight values from the compressed data. As typical decompression procedures are serialized by definition, the software-based decompression significantly increases the overall processing latency. For obtaining meaningful performance improvements, hence, the software approaches require extremely sparse data that cannot be provided by the practical pruning ratios for transformers, e.g., less than 5% sparsity is required for enhancing the system performance using CSR-encoded data even utilizing the sparsity-aware processing kernels of NVIDIA GPGPUs [13]. Targeting the practical transformer pruning ratios of less than 0.8, the hardware-based decompression units should be installed at the memory interface architecture [2]. Although the hardware approaches greatly accelerate the decompression steps, it is hard to fully utilize the given physical memory bandwidth, i.e., the processing engines cannot seamlessly access the compressed model from the memory. More precisely, the sparse matrices in the pruned transformers generally have random patterns, and the number of unpruned weights in the compressed data cannot be deterministic by nature. If we adopt the weight-oriented compression methods like CSR or its variations [1], the memory accessing patterns become irregular depending on the pruning patterns, requiring additional model accessing cycles associated with memory stalls [2].

The recent XOR-gate compression, which is the baseline compression method in this work, targets the regular accessing patterns of compressed weights by introducing fixed-in and fixed-out decompression tables as summarized in Fig. 1 [7, 14]. Note that the sparse original matrix is compressed by a shared look-up table (LUT) that can be realized by only using XOR gates. Due to the limited representable symbols, however, the XOR-gate compression requires to add extra data to correct the mis-recovered bits, which are denoted as patches in Fig. 1. As the number of errors per encoded vector varies depending on the local sparsity [2], the memory requests for patches become irregular, causing unwanted latency overheads for constructing valid weights to be computed at the following processing engines. The recent studies may reduce these extra cycles by utilizing sufficiently-large on-chip buffers [2] or by ignoring less important patches [10]. However, existing approaches basically depend on the target pruned transformer, which cannot be the generalized solution to achieve the ideal model-accessing speed for the given physical memory bandwidth.

2.2 Effective Bandwidth

To quantitatively evaluate the accessing speed of pruned transformers, the concept of effective bandwidth is newly defined as

$$eBW = \left(1 - \frac{s_{\text{extra}}}{s_{\text{weight}} + s_{\text{extra}}}\right) \times \left(1 - \frac{n_{\text{stall}}}{n_{\text{req}} + n_{\text{stall}}}\right) \times \frac{BW}{(1-p)} = \alpha \times \beta \times eBW_{\text{ideal}}. \quad (1)$$

We first derive the ideal model-accessing speed of the pruned transformer based on the given physical memory bandwidth (BW) and the pruning ratio (p), i.e., $eBW_{\text{ideal}} = BW/(1-p)$, indicating that the decompression steps reconstruct the dense matrices from the unpruned weights stored in the memory [2]. Two degrading factors α and β are also introduced to calculate the effective bandwidth, where $0 \leq \alpha, \beta \leq 1$. Observed by the processing engines, the first degrading factor α denotes the bandwidth degradation caused by the non-ideal extra data in the target compressed data format. More specifically, the compressed data of the pruned transformer generally consists of two parts: 1) compressed data related to the unpruned weights, and 2) extra data indicating the pruned position or the valid weight reconstruction, where the corresponding data sizes are denoted as s_{weight} and s_{extra}, respectively [7]. For the original model size of s_{org}, the recent compression formats normally reduce s_{weight} to the ideal value for the given pruning ratio p, i.e., $s_{\text{weight}} = s_{\text{org}} \times p$. However, s_{extra} strongly depends on the selected compression format, e.g., coordinate information for CSR-encoded data [1] and data correction patches for XOR-gate compression [7, 14]. As described in (1), utilizing more extra data directly reduces the α-factor, decreasing eBW far from the ideal value. For the second degrading factor β, we consider the number of memory requests for accessing the compressed data, denoted as n_{req}. In addition, for the first time, we carefully investigate the number of memory stalls, i.e., n_{stall}, mainly caused by the reduced utilization of the physical memory bandwidth. When there exist numerous irregular memory access patterns on weights or extra data, the decompression procedures need to synchronize all the required data for the current compressed block, forcing idle periods at memories and keeping regular data [14]. As shown in (1), the

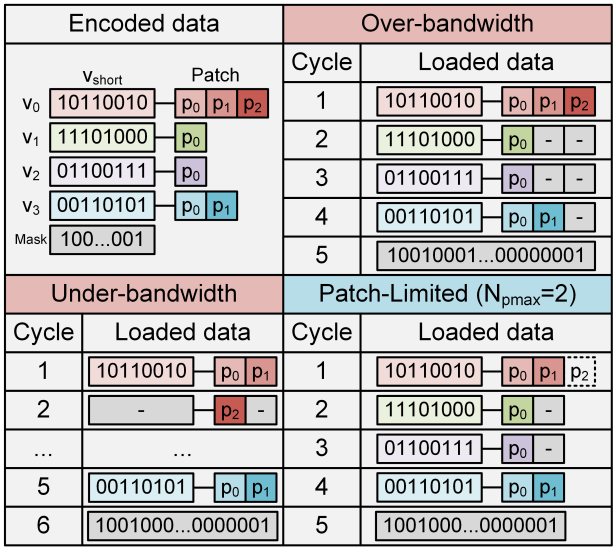

Figure 2: Encoded Data Loading Scenario for Different XOR-gate Compression Types and Physical Bandwidth

stall cycles obviously reduce the β-factor, accordingly sacrificing the effective bandwidth observed by the processing engines.

In order to achieve the high-performance transformer acceleration system, therefore, it is required to minimize two degrading factors for increasing eBW as close as to eBW_{ideal}, eventually providing enough model-accessing speed even for the pruned transformers [18]. In the previous studies for storing pruned transformers, advanced compression formats have continuously developed to reduce the value of α by minimizing the size of extra data s_{extra}. However, studies for reducing β are rarely reported, especially for the transformer-based applications. Some recent solutions with XOR-gate compression formats from [2] may reduce the number of irregular accesses compared with the conventional CSR-based approaches. However, even for the state-of-the-art solutions, it is impossible to totally remove the memory stalls causing the unwanted bandwidth degradation, e.g., the vertical patch storing only reports $\beta = 0.55$ for the case study of 0.6-pruned transformer model on WMT 2014 English-German dataset, mainly caused by the irregular number of patches per encoding vector [2]. In this work, the stall-free memory access for pruned transformers is developed for the first time by modifying the previous XOR-gate compression format, while additional optimizations are followed to gradually increase β and α values for maximizing the effective bandwidth.

3 OPTIMIZING XOR-GATE COMPRESSION

3.1 Patch-Limited XOR-gate Compression

In the previous algorithms that aim for lossless compression [7, 14], all the patch data were required to correct the original model. However, as the number of XOR-gate compression errors differs from vector to vector, it is not easy to prepare all the patch data for the encoding vector. In the case of an over-bandwidth system

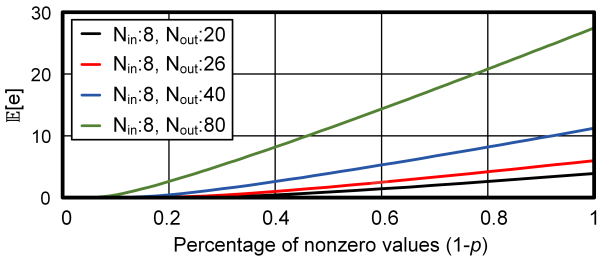

Figure 3: Expected Error with Sparsity and N_{out} Change.

targeting a maximum number of patches, as depicted in the upper-right of the Fig. 2, redundant dummy data increase s_{extra}, degrading α-factor. On the other hand, if the system bandwidth is lower than the maximum patch case, as illustrated in the lower left of the figure, the system meets the out-of-patch situation, which leads to an increased n_{stall} and reduces β-factor.

To create a stall-free system that targets the eBW_{ideal}, instead, we propose Patch-Limited (PL) XOR-gate Compression. The proposed PL XOR-gate compression limits the maximum number of patch data by partially skipping error correction. In the lower-right of the Fig. 2, the maximum number of patches, or N_{pmax}, is set to 2. As the third patch p_2 is ignored, with the same physical bandwidth as the under-bandwidth case, the PL case can avoid memory stall without decreased β-factor. Namely, the proposed PL XOR-gate compression guarantees the upper bound of the model with the irregular data-accessing pattern. However, as N_{pmax} and s_{extra} becomes smaller, the model quality also decreases. This is due to the large number of skipped error corrections caused by their large number and unbalanced distribution. In the following section, we introduce the Partially-Structured Transformer pruning technique that can solve both of the patch amount and distribution problem.

3.2 Partially-Structured Transformer Pruning

To deal with the large patch and distribution problem, we started by investigating the relation between the sparsity of the encoding vector and error. Fig. 3 displays the expected number of errors when compressing N_{out} bits of sparsity p down to N_{in} bits. The compressed data, with a size of $N_{\text{in}} = 8$, is optimized for XOR-gate compression using two shift registers, as described in [14]. As N_{in} is bounded by the compression complexity [14], we can control p by changing N_{out}. The expected number of errors $\mathbb{E}[e]$ converges to zero when the proportion of nonzero values $(1 - p)$ falls below the compression ratio $\frac{N_{\text{in}}}{N_{\text{out}}}$ [14]. For instance, the green line represents a pruned model with sparsity $p = 0.9$. That is, the expected number of errors increases when more than 10% of the values are nonzero in the case of $p = 0.9$.

To regulate the number of nonzeros in the encoding vector, we propose the Partially-Structured Transformer (PST) pruning. Fig. 4 shows two types of pruning techniques and corresponding XOR-gate compression results. The numbers in the dense matrix represent the importance score of each weight. In the case of fine-grained pruning, weights with low importance scores are removed regardless of their position. As the importance score is evaluated in the global scope, the survival weights' position can be concentrated.

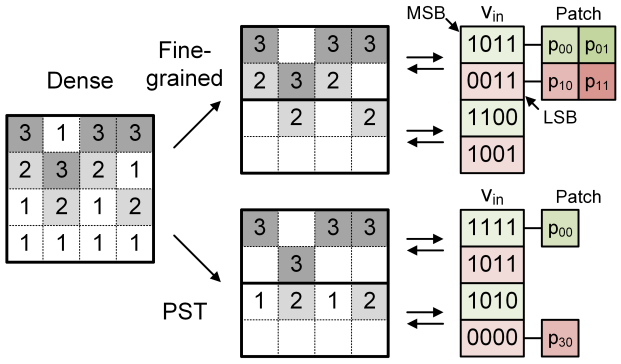

Figure 4: Fine-grained Pruning and PST Pruning Example.

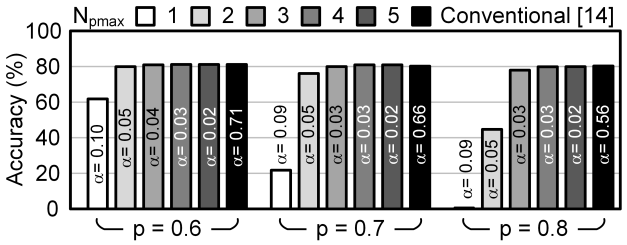

Figure 5: DeiT-base Model, ImageNet Classification top-1 Accuracy for Different Pruning Methods and Patch Limit

The biased survived weights pattern generates local sparsity fluctuation, which leads to a large number of patches. In the upper right of the figure, the locally dense upper side of the matrix generates four patches, while the locally sparse lower side of the matrix makes no patch. On the contrary, the proposed PST pruning limits the local sparsity, targeting uniform local sparsity like NVIDIA N:M pruning [12]. For this, the PST pruning first divides the model into groups with N_{out} weights. After that, fine-grained pruning is applied inside the groups. Unlike N:M pruning, the group size of proposed PST pruning targets XOR-gate compression encoding vector size N_{out}, which is not a power of 2. As the group size of PST is much larger than N:M pruning, the increased freedom of the nonzero weight pattern effectively helps the model to preserve accuracy. The acquired model with normalized local sparsity relaxed the patch distribution and decimated the number of total patches.

3.3 Bit-wise Patch Reduction

Although previously introduced PL XOR-gate compression and PST pruning successfully achieved stall-free decoding and normalized patch distribution, redundant patch data is still increasing s_{extra}. As denoted in Fig. 1, the original PL XOR-gate compression loads all the patches regardless of the significance of weight bits. However, as reported in PEC [10], the importance of the patch does exist. Namely, the patch for the MSB vector has to be prioritized compared to the one for the LSB vector. Unfortunately, the PEC [10] algorithm was too time-complex and hard to apply on the recent large models, due to the iterative evaluation process. In the proposed Bit-wise Patch Reduction (BPR), instead, evaluate the significance simply by its bit position. That is, BPR only corrects errors in selected bits while skipping the error correction in others. This is possible because most of the errors are already decimated using PST pruning. In practice, in the case of DeiT-base for the ImageNet classification task and BERT-base for SQuAD v1.1, correcting the top two MSBs was enough to preserve the original model quality. Furthermore, the reduced number of errors enabled the required number of patches to become less than one, with negligible cost. In other words, to minimize the s_{extra}, we allowed only one error correction for a group of consecutive encoding vectors. Thanks to the BPR, we could reduce the patch overhead by more than 99% on average, which led to increased α and eBW.

4 EXPERIMENTAL RESULTS

4.1 Testing Environments

To evaluate the impact of the proposed work fairly, we conducted tests on two different Transformer models and tasks: 1) DeiT-base [16] model for ImageNet classification, 2) BERT-base [3] model for SQuAD v1.1 [15]. They were started from the pre-trained model and then pruned using magnitude pruning, retrained for 100 epochs, and subsequently had weight-only quantization applied in the INT8 format for encoder and decoder weights. The total number of encoder/decoder parameters is 85M for both models. To compress the pruned model with N:M sparsity, XOR-gate compression, and stacked XOR-gate compression, we employed the following techniques: 1) the XOR-gate compression with shift register [14], 2) the previous State-of-the-art XOR-gate compression memory interface [2], 3) NVIDIA N:M sparsity [12]. The N:M sparsity used the following configuration: $(p, N) = \{(0.6, 4), (0.7, 3), (0.8, 2), (0.9, 1)\}$, $M = 10$. For the XOR-gate compression, the following configuration is used: $N_{\text{out}} = 8$, $(p, N_{\text{out}}) = \{(0.6, 20), (0.7, 26), (0.8, 40), (0.9, 80)\}$, number of shift register $N_s = 2$. The VA-patch memory interface [2] is used to evaluate the effective bandwidth of conventional XOR-gate compression. The physical bandwidth is set to 930GBps, which is the same as the previous work [2], for a fair comparison.

4.2 PL XOR-gate Compression and α-factor

Achieving stall-free operation, the proposed PL XOR-gate compression is evaluated on the DeiT-base model. Fig. 5 shows the compressed model's accuracy and α-factor, applying different patch limit N_{pmax} on the DeiT-base model with ImageNet dataset. The first white box with $N_{pmax} = 1$ case loads one patch data per encoding vector, which already caused large size of s_{extra}. Nevertheless, due to the XOR-gate compression error, the quality of the model decreased severely, which is hard to use in practice. As we increase N_{pmax}, the accuracy is recovered, but now the α-factor decreases, which leads to a reduced eBW. Even the smallest patch limit $N_{\text{pmax}} = 1$ case degrades α-factor into less than 0.1, which is much worse than the conventional XOR-gate compression. This is because of the excessive size of s_{extra}. In the conventional XOR-gate compression, most of the encoding vector has no patch. However, due to the PL XOR-gate compression, the encoding vector is required to load a large number of patches, including a dummy patch, which is much larger than the real patch. The rightmost black box, which represents the conventional XOR-gate compression, has a

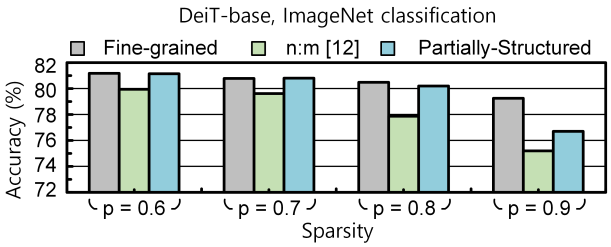

Figure 6: Accuracy Comparison for Different Pruning.

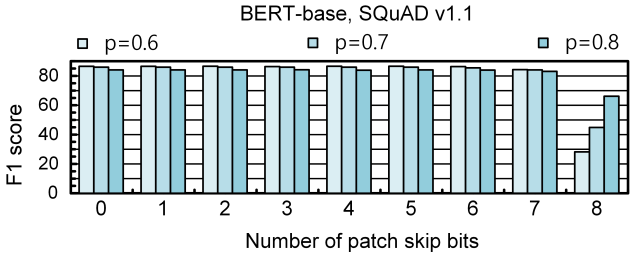

Figure 7: Result for Bit-wise Patch Skip

Table 1: Pruning Performance Comparison

Metric	Method	S=0.6	S=0.7	S=0.8
DeiT-base (total patch)	Fine-grained	2,900K	2,295K	2,412K
	PST (ours)	**69,437**	**55,371**	**34,100**
BERT-base (total patch)	Fine-grained	2,101K	1,735K	1,946K
	PST (ours)	**68,076**	**50,541**	**30,053**
DeiT-base (variance)	Fine-grained	0.096	0.104	0.183
	PST (ours)	**0.002**	**0.002**	**0.003**
BERT-base (variance)	Fine-grained	0.069	0.080	0.160
	PST (ours)	**0.002**	**0.002**	**0.002**

Table 2: Result for Different Compression Techniques

Model	Method	α	β	eBW (Tbps)
DeiT-base (S=0.6)	XOR	0.71	0.55	0.9
	sXOR	0.66	0.89	1.36
	eBW_{ideal}	1	1	2.32
Deit-base (S=0.7)	XOR	0.66	0.54	1.11
	sXOR	0.62	0.87	1.66
	eBW_{ideal}	1	1	3.09
DeiT-base (S=0.8)	XOR	0.56	0.75	1.94
	sXOR	0.54	0.92	2.29
	eBW_{ideal}	1	1	4.64
BERT-base (S=0.8)	XOR	0.56	0.75	1.95
	sXOR	0.55	0.89	2.29
	eBW_{ideal}	1	1	4.64

relatively high α-factor, but it suffers from an irregular memory access pattern. To minimize the s_{extra} while preserving the model quality and stall-free operation, we further applied the proposed PST and BPR in the following experiments.

4.3 PST Pruning and Patch Distribution

Fig. 6 illustrates the accuracy of the ImageNet classification task across various pruning methods and the levels of sparsity. Compared to fine-grained pruning, the proposed PST pruning shows less than 0.5% accuracy drop in the practical sparsity range. We also tested NVIDIA N:M sparsity technique [12] on the same network. However, the group size targeted by the proposed PST pruning, N_{out}, is 2-8 times larger than that of the N:M technique. Therefore, the model with NVIDIA N:M sparsity suffered from a severe accuracy drop due to the limited freedom for the survived weight pattern, which differs from the model with proposed PST pruning.

Having confirmed that PST has minimal impact on accuracy, we now turn our attention to the effect of XOR-gate compression errors and their distribution. For the two transformer-based models, Table 1 displays the total number of errors and their variance of the model. As the number of nonzero values in each encoding vector is limited, most errors are removed, showing less than 3% of the patch left for both models. The patch distribution also normalized, thanks to the PST. The lower part of the table shows a relaxed distribution of patches. In the conventional XOR-gate compression, a large number of errors occurred in the locally dense vectors. However, the normalized number of nonzero weights in encoding vectors successfully removed the patch distribution problem. Based on the reduced number of errors, we moved on to the BPR, which targets to recover decreased α-factor by PL XOR-gate compression.

4.4 Bit-wise Patch Reduction Analysis

In order to reduce the s_{extra}, we started to test the error resiliency of the Transformer model, varying the number of patch skip bits. Fig. 7 illustrates the F1 score of the BERT-base model with PST pruning for different sparsity. The X-axis represents the number of patch skipped bits, starting from LSB to MSB. The "0" represents a fully reconstructed model performance, the same as the model with PST pruning. The F1 score drops by less than 0.5% when error correction skips up to six bits. However, when seven bits are skipped, the F1 score decreases. Finally, when all the patches are skipped, the F1 score of the reconstructed model experienced a significant drop. The DeiT-base model ImageNet classification accuracy shows a similar trend, but we opted for the SQuAD task due to space constraints.

Based on the erroneous model performance analysis, we concluded that correcting at least two MSBs is critical to the model performance. Also, considering the greatly reduced number of errors, we even reduced the N_{pmax} under 1 to minimize the s_{extra} suffered from the dummy patch. As the memory interface is based on the VA-patch [2], the proportion of the patch could be less than one by loading one patch for multiple encoding vectors. Table 2 shows α, β-factor, accuracy, eBW of transformer models varying sparsity p and compression techniques. As stacked XORNet [2] reduced patch distribution, despite of the reduced s_{extra}, increased β-factor achieved a higher eBW. The proposed PL, PST, and BPR successively solved the β-factor, accuracy, and α-factor problems

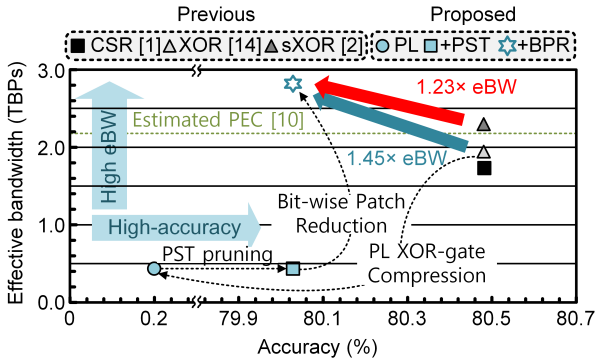

Figure 8: Investigating the Accuracy-eBW Trade-offs to Support the DeiT-base Model for p=0.8.

to achieve the highest *eBW*. The same trend was observable in different sparsity and models, achieving 23% *eBW* gain on average. Note that the proposed work shows higher *eBW* than PEC [10], which is estimated from the reported model size.

Fig. 8 illustrates the accuracy-effective bandwidth trade-offs by adopting different compression methods for the DeiT-base model with $p = 0.8$. The conventional XOR-gate compression and stacked XORNet suffered from the irregular data pattern, achieving *eBW* far from the eBW_{ideal}. Therefore, we first introduced the novel Patch-Limited XOR-gate compression, enabling stall-free memory access. However, due to the large amount of skipped error correction, the model quality severely decreased, denoted as a blue circle. Then, we proposed Partially-Structured Transformer pruning, which decimated more than 97% of the total patch. Lastly, Bit-wise Patch Reduction removed most of the dummy patches, exceeding the *eBW* of the State-of-the-art compression method [2]. The *eBW* of approximated PEC [10] is also noted as a green dashed line, but the proposed work shows higher *eBW* thanks to the stall-free memory access. The proposed compression method significantly increased *eBW* by 23% compared to the previous work [2], with less than 0.5% accuracy loss.

5 DISCUSSION

In this work, we introduced innovative compression techniques tailored for practical large models adept at compressing transformer models using XOR-gate compressed data. The introduced novel Patch-Limited XOR-gate compression successfully removed memory stalls by limiting the maximum number of error corrections per vector. Furthermore, to recover the reduced α-factor, we introduced a Partially-Structured Transformer pruning targeting efficient XOR-gate compression by decimating most of the error. Lastly, redundant patches are removed by allocating Bit-wise Patch reduction to the six LSBs. Compared with the previously introduced stacked XORNet, the experimental result shows that the proposed work increased *eBW* more than 23% while sacrificing negligible accuracy.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) (No. 2022R1A2C2092521, RS-2023-00258227), Institute of Information & communications Technology Planning & Evaluation (IITP) (No.RS-2023-00229849), with all three grants funded by the Korea government's Ministry of Science and ICT (MSIT).

REFERENCES

[1] Aydin Buluç, Jeremy T Fineman, Matteo Frigo, John R Gilbert, and Charles E Leiserson. 2009. Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. In *Proceedings of the 21th Annual Symposium on Parallelism in Algorithms and Architectures*. 233–244.

[2] Younghoon Byun, Seungsik Moon, Baeseong Park, Se Jung Kwon, Dongsoo Lee, Gunho Park, Eunji Yoo, Jung Gyu Min, and Youngjoo Lee. 2023. Sparsity-Aware Memory Interface Architecture using Stacked XORNet Compression for Accelerating Pruned-DNN Models. *Proceedings of Machine Learning and Systems* 5 (2023).

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[4] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

[5] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[6] Eldar Kurtic, Elias Frantar, and Dan Alistarh. 2023. ZipLM: Inference-Aware Structured Pruning of Language Models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[7] Se Jung Kwon, Dongsoo Lee, Byeongwook Kim, Parichay Kapoor, Baeseong Park, and Gu-Yeon Wei. 2020. Structured compression by weight encryption for unstructured pruning and quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1909–1918.

[8] Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. 2022. A fast post-training pruning framework for transformers. *Advances in Neural Information Processing Systems* 35 (2022), 24101–24116.

[9] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *arXiv preprint arXiv:2309.06180* (2023).

[10] Hyunseung Lee, Jihoon Hong, Soosung Kim, Seung Yul Lee, and Jae W Lee. 2023. A Memory-Efficient Edge Inference Accelerator with XOR-based Model Compression. In *Proceedings of the 60th ACM/IEEE Design Automation Conference (DAC)*. 1–6.

[11] Jung Gyu Min, Dongyun Kam, Younghoon Byun, Gunho Park, and Youngjoo Lee. 2023. Energy-Efficient RISC-V-Based Vector Processor for Cache-Aware Structurally-Pruned Transformers. In *2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 1–6.

[12] Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. 2021. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378* (2021).

[13] Maxim Naumov, L Chien, Philippe Vandermersch, and Ujval Kapasi. 2010. Cusparse library. In *GPU Technology Conference*.

[14] Bae Seong Park, Se Jung Kwon, Daehwan Oh, Byeongwook Kim, and Dongsoo Lee. 2022. Encoding Weights of Irregular Sparsity for Fixed-to-Fixed Model Compression. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Vs5NK44aP9P

[15] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).

[16] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*. PMLR, 10347–10357.

[17] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[18] Shikhar Tuli and Niraj K Jha. 2023. AccelTran: A sparsity-aware accelerator for dynamic inference with transformers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2023).

[19] Eunji Yoo, Gunho Park, Jung Gyu Min, Se Jung Kwon, Baeseong Park, Dongsoo Lee, and Youngjoo Lee. 2023. TF-MVP: Novel Sparsity-Aware Transformer Accelerator with Mixed-Length Vector Pruning. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.