DeepCompress-ViT: Rethinking Model Compression to Enhance Efficiency of Vision Transformers at the Edge

Sabbir Ahmed[1*], Abdullah Al Arafat[2*], Deniz Najafi[3*], Akhlak Mahmood[4], Mamshad Nayeem Rizve[5], Mohaiminul Al Nahian[1], Ranyang Zhou[3], Shaahin Angizi[3], Adnan Siraj Rakin[1]
[1]Binghamton University (SUNY), [2]North Carolina State University,
[3]New Jersey Institute of Technology, [4]Georgia Institute of Technology, [5]Adobe Inc.

Abstract

Vision Transformers (ViTs) excel in tackling complex vision tasks, yet their substantial size poses significant challenges for applications on resource-constrained edge devices. The increased size of these models leads to higher overhead (e.g., energy, latency) when transmitting model weights between the edge device and the server. Hence, ViTs are not ideal for edge devices where the entire model may not fit on the device. Current model compression techniques often achieve high compression ratios at the expense of performance degradation, particularly for ViTs. To overcome the limitations of existing works, we rethink model compression strategy for ViTs from first principle approach and develop an orthogonal strategy called ***DeepCompress-ViT****. The objective of the DeepCompress-ViT is to encode the model weights to a highly compressed encoded representation using a novel training method, denoted as Unified Compression Training (UCT). Proposed UCT is accompanied by a decoding mechanism during inference, which helps to gain any loss of accuracy due to high compression ratio. We further optimize this decoding step by reordering the decoding operation using associative property of matrix multiplication, ensuring that the compressed weights can be decoded during inference without incurring any computational overhead. Our extensive experiments across multiple ViT models on modern edge devices show that DeepCompress-ViT can successfully compress ViTs at high compression ratios ($> 14\times$). DeepCompress-ViT enables the entire model to be stored on edge device, resulting in unprecedented reductions in energy consumption ($> 1470\times$) and latency ($> 68\times$) for edge ViT inference. Our code is available at https://github.com/ML-Security-Research-LAB/DeepCompress-ViT.*

1. Introduction

The integration of computer vision and deep learning (DL) into the Industrial Internet of Things (IIoT) and edge devices has revolutionized data processing, analysis, and action in these systems [3]. By deploying advanced models, such as Vision Transformers (ViTs), on resource-constrained devices, IIoT systems can directly manage complex tasks, including object detection, anomaly detection, and facial recognition on the device. This local processing capability on edge minimizes data transmission to centralized servers, reducing latency and enhancing data privacy by keeping information closer to its source [35].

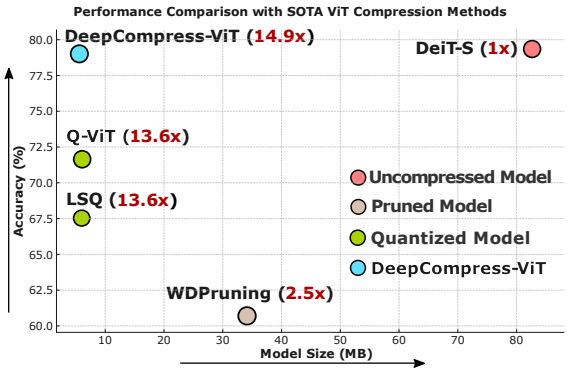

Figure 1. *Comparison of DeepCompress-ViT with SOTA ViT compression methods [8, 11, 31] for compressing DeiT-S [27] trained on Imagenet-1k [22]. Existing compression methods achieve a large compression ratio at the expense of large performance loss, whereas the DeepCompress-ViT achieves the highest compression ratio with the best accuracy among the competition.*

In recent years, Vision Transformers (ViTs) [7] have gained significant attention among deep neural network (DNN) models for their remarkable performance [20]. Their potential to tackle various real-world vision problems, such as object detection [4] and segmentation [26], makes them especially appealing for edge applications. However, the challenge of achieving reliable, real-time inference on resource-constrained devices cannot be underestimated, primarily due to the large model sizes of ViTs in comparison to models like Convolutional Neural Networks [23].

The architectural design of ViTs often exceeds the on-chip memory limits of edge devices (e.g., Espressif ESP32-

* These authors contributed equally

S3 [1] has 8 MB pseudo-static RAM on-chip memory, but DeiT-S model size is approximately 85 MB), leading to a reliance on off-chip memory from servers to store the entire model. This dependency introduces two critical challenges that significantly impede the use of large ViTs in edge environments [10]. Firstly, computations performed on-chip are estimated to be 10 to 100 times faster than those conducted off-chip (see Table 1 for a detailed numerical comparison of various metrics between on- and off-chip memories). Secondly, transferring sizable model weights demands high communication bandwidth, resulting in elevated energy consumption. As a result, for edge applications (e.g., real-time patient monitoring, autonomous vehicle perception, quality control in production) where latency and power efficiency are critical, the deployment of ViT models is severely limited. *Therefore, addressing model size and enhancing memory management are not just challenges; they are essential for unlocking the full potential of ViT models in IoT devices.*

To effectively deploy large Vision Transformer (ViT) models at the edge, it is crucial to aggressively compress them to fit into on-chip memory. Aggressive compression reduces reliance on off-chip memory, decreasing latency and energy consumption. However, techniques such as pruning [31] and quantization [11] often face challenges at high compression ratios, leading to significant drops in performance. For instance, the DeiT-Small model, which is approximately 85 MB, would require over 10× compression to fit into an 8 MB edge device. This extreme compression level can result in severe accuracy loss (please refer to Figure 1). *Therefore, it is essential to carefully reconsider and rethink model compression strategies for ViTs to maintain both performance and efficiency at the edge.*

In this work, we begin by formulating the popular compression strategies from the perspective of weight encoding, as illustrated in Figure 2. Both pruning and quantization are techniques that encode model weights into a compressed format. However, at high compression ratios, this weight encoding mechanism can become lossy, leading to a decline in model performance. Our hypothesis is that to mitigate the performance loss due to encoded weights, it is essential to implement a decoding method during inference that can reconstruct or remap the weights back to their original (decompressed) state. This approach could enhance the model's accuracy during inference by reversing the effects of a lossy compression. Nonetheless, developing an effective compression strategy that achieves high compression while also enabling accurate weight decoding for inference involves overcoming several challenges, which our work aims to resolve.

To this end, we propose a novel and unique perspective on model compression, which enables aggressive compression of ViT models while maintaining a decoding framework to reconstruct the weights back. The key idea is that the compressed version of the weight and a lightweight decoder will be stored in memory to reduce memory costs. However, these weights will be reconstructed back to their original form using the decoder in a layer-by-layer sequence for model inference and thus help to maintain accuracy. Our proposed ***DeepCompress-ViT*** consists of two phases—*Unified Compression Training* and *Optimized Test-Time Decoding*. In Unified Compression Training, we train an encoder and decoder alongside the model, learning a highly compressed representation of its parameters through encoded embeddings. After training, we retain only these encoded weights and the decoder, discarding the encoder.

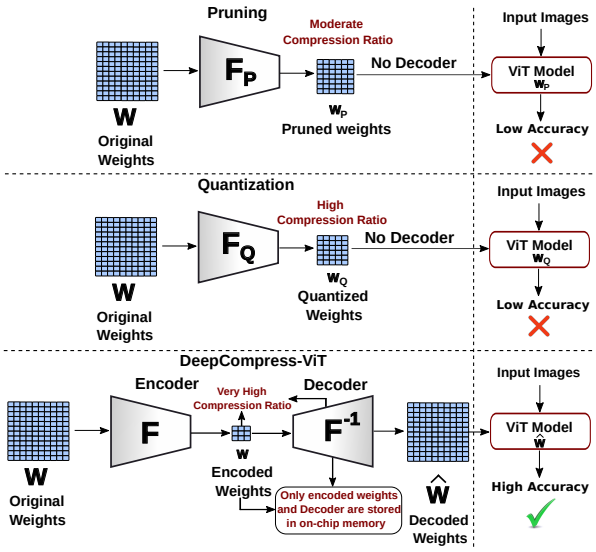

Figure 2. *Existing compression methods can be viewed as a form of weight encoding. However, inference with lossy encoded weights results in performance dorp whereas proposed DeepCompress-ViT maintains an optimized decoding method at test time to maintain model performance.*

Next, during inference, the goal is to Decode the compressed parameters layer-by-layer systematically to reconstruct the original parameters for each layer to compute the next layers output. Nevertheless, the proposed layer-by-layer decoding approach comes with two notable drawbacks. First, decoding the compressed representation of each layer back to its original form incurs extra computational costs, such as increased energy use and processing time. Additionally, this can lead to significant on-chip memory usage, particularly in larger layers, when decoded weights will be stored on-chip temporarily to compute output of a given layer. Hence, we propose *Optimized Test-Time Decoding*, eliminating the need to decode embeddings to their original parameters. This optimized decoding step leverages the associative property of matrix multiplication to reorder the decoding operation. This innovation eliminates both computation and memory usage drawback associated with the decoding step, making proposed

DeepCompress-ViT best of all worlds (accuracy, memory, energy and latency). Our experiments on SOTA ViTs and edge devices show that *DeepCompress-ViT* achieves compression ratios exceeding 14× while achieving unprecedented reductions in energy ($> 1470\times$) and latency ($> 68\times$), all with the least amount of accuracy loss among the SOTA compression methods.

2. Related Work

Vision Transformers. The introduction of Transformer architecture [28] has marked a paradigm shift in machine learning, initially transforming natural language processing. Building on this success, the architecture was adapted for computer vision, leading to the development of Vision Transformer (ViT) [7]. This success has led to a surge in transformer-based models for computer vision. For example, DeiT [27] introduced advanced training techniques and a novel distillation approach to reduce ViT's dependence on large datasets. Further architectural innovations, such as Pyramid Vision Transformer (PVT) [29] and Swin Transformer [17], incorporated hierarchical feature representations enabling transformers to serve as versatile frameworks for diverse computer vision applications. Even though ViTs have demonstrated strong performance, they require significant memory that limits their application in edge devices. Compressing ViTs has thus emerged as an important area of research to address this large memory requirement.

ViT Compression. A widely used approach for compressing Vision Transformers (ViT) is pruning [16, 30–33, 36], which focuses on reducing the number of unimportant weights in the model. Pruning methods for ViT models are commonly divided into two categories: unstructured and structured pruning. Unstructured pruning techniques, such as magnitude-based and Hessian-based methods [24], remove individual weights based on certain importance metrics. However, these techniques result in irregular sparsity patterns, which complicate tensor computations and limit hardware efficiency. Structured pruning mitigates this challenge by removing larger, contiguous structures within the model, such as entire attention heads or rows of weight matrices, which are more compatible with hardware requirements. For instance, [31] proposed a structured pruning method for ViTs that uses a binary mask to distinguish important from unimportant parameters based on magnitude. Another effective approach, introduced in [33], combines pruning, layer skipping, and knowledge distillation within a unified framework to create a compact ViT model.

Another widely adopted approach for compressing ViT models is quantization [8, 11–13, 15, 18], which converts floating-point parameters to lower precision values. Several quantization methods tailored to ViT's unique architecture have been proposed. For instance, Q-ViT [11] introduced differentiable quantization for ViTs, treating quantization bit-widths and scales as learnable parameters. PTQ4ViT [18] utilizes twin uniform quantization and applies a Hessian-guided metric to determine optimal scaling factors. FQ-ViT [15] incorporates powers-of-two scale quantization and log-int quantization specifically for LayerNorm and Softmax operations, respectively.

While both pruning and quantization methods have shown great promise in reducing computational demands, they still experience significant performance degradation at high compression ratios [11, 31], also shown in Table 2, severely limiting their applicability in edge IoT devices.

3. Challenges and Motivation

To motivate our compression technique, we identify the critical bottleneck hindering efficient inference of Vision Transformers (ViTs) at the edge. We will also assess the limitations of current compression methods, establishing the necessary foundation for our proposed method.

3.1. Challenges of Efficient Edge Inference

The deployment of ViTs on edge devices presents significant challenges due to their substantial memory requirements, which often exceed the on-chip memory capacity of these resource-constrained environments. Consequently, model parameters must be stored in off-chip memory, necessitating a sequential processing approach during inference. This approach involves iteratively loading each layer's weights into on-chip memory and performing computations on that layer before proceeding to the next. While this method enables the inference of ViTs on devices with limited on-chip memory, it introduces significant latency and increased energy consumption due to frequent off-chip memory accesses, both of which are critical concerns in resource-constrained edge environments (e.g., autonomous vehicle perception).

Table 1. *Comparison of On-chip and Off-chip Memory Access Cost in Edge Devices*

Memory Type	Access Time (ns)	Energy per Access (pJ)	Available Storage (MB)
On-chip	1-10	5-20[†]	1-8
Off-chip	70-200	1300-2600[§]	1024-8192[*]

[†] 32b SRAM Read (32KB), [§] 32b DRAM [10], * For edge AI devices.

One potential solution to mitigate this frequent off-chip memory access is model compression. Model compression can reduce the overall memory footprint and allow the entire model to be stored in on-chip memory, which can minimize/eliminate the need for costly memory access. However, as shown in Table 1, the available on-chip memory is often severely constrained (e.g., $\sim$ 1-8 MB), and achieving this would require aggressive compression (e.g., greater than 10×).

One of the widely explored approaches for model compression is pruning. Numerous studies [24, 31, 33] have applied pruning techniques on ViTs to reduce their memory and computation costs. While pruning has demonstrated significant success in lowering the computational demands

of these models, our investigation reveals that achieving high compression through pruning in ViTs is challenging without severely impacting model performance. As shown in Table 2, even with a 2.5 $\times$ compression ratio, the model suffers $\sim$ 19 % accuracy loss.

Another widely adopted approach for improving model efficiency is quantization [11, 15, 18]. Quantization has proven particularly effective in reducing the memory footprint of large models, such as large language models (LLMs). The existing literature demonstrates that extreme quantization can be successfully applied to LLMs [19] with minimal performance loss. However, quantization applied to ViTs presents a different challenge. Studies focusing on ViTs [11, 34] have shown that these models suffer from significant performance degradation when subjected to extreme quantization (see Table 2).

Table 2. *Evaluating existing compression methods by aggressively compressing a DeiT-S model and highlighting their shortcoming in achieving our ideal objective (4^{th} row) for edge inference.*

Method	Original Size (MB)	Compression Ratio	Accuracy (%)
Original Model	84.1	1×	79.72
Pruning [11]	33.7	2.5×	60.51
Quantization [31]	6.18	13.6×	71.90
Our Objective	1-8 (constraint at edge)	10-80× (required)	79.72 (ideally)

Next, we dissect these two popular model compression strategies to determine the fundamental limitation that hurts the state-of-the-art model compression technique and whether we can develop an orthogonal approach to overcome them at our targeting edge applications.

3.2. Motivation

As shown in Figure 2, both pruning and quantization can be viewed as a form of encoding original model weights to a compressed representation. The fundamental challenge lies in the way these compression techniques encode model weights. In this work, we postulate that the primary factor contributing to this performance loss can be attributed to the large weight perturbation error introduced by the encoding mechanism of these compression techniques, as formalized in the following lemma.

Lemma 3.1. ***(Error Bound for Compression in DNNs).*** *Let $\mathbf{w}$ represent the weights of a deep neural network (DNN) with an associated loss function $\mathcal{L}(\mathbf{w})$. Consider a compression technique that encodes the weights using a transformation $\mathbf{F}$ resulting in $\mathbf{w}' = \mathbf{F}(\mathbf{w})$ and introducing a perturbation $\delta\mathbf{w} = \mathbf{w}' - \mathbf{w}$. The change in loss due to this compression can be upper bounded as follows:*

$$|\mathcal{L}(\mathbf{w}') - \mathcal{L}(\mathbf{w})| \leq \|\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w})\|\|\delta\mathbf{w}\| + \frac{1}{2}\|H\|\|\delta\mathbf{w}\|^2,$$

where $\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w})$ is the gradient of the loss function with respect to the original weights $\mathbf{w}$, $\|H\|$ denotes the norm of the Hessian matrix of $\mathcal{L}(\mathbf{w})$, and $\|\delta\mathbf{w}\|$ is the norm of the weight perturbation introduced by the compression.

From Lemma 3.1, it is evident that a smaller weight perturbation $\|\delta\mathbf{w}\|$ leads to a tighter upper bound on the loss difference $|\mathcal{L}(\mathbf{w}') - \mathcal{L}(\mathbf{w})|$. Therefore, an ideal compression method should minimize the weight perturbation error. However, for edge application of ViTs, we are forced to perform aggressive compression that will inevitably lead to a higher $\|\delta\mathbf{w}\|$.

Neither pruning nor quantization has a countermeasure against this high $\|\delta\mathbf{w}\|$ for aggressive compression, laying down the foundation for our proposed method as to whether we can device a technique to reduce $\|\delta\mathbf{w}\|$ at the inference stage even at a high compression ratio. The research question that motivates our work: *Given that weight encoding for compression inevitably introduces significant weight perturbation error at high compression ratio, an effective approach to mitigate this error may lie in coupling the encoding mechanism with a complementary decoding mechanism that can reverse the lossy encoding*, as formalized in the following proposition.

Proposition 3.2. ***(Lower Error Bound with Weight Decoding).*** *Let $\mathbf{w}$ represent the weights of a deep neural network (DNN) and consider a compression method that encodes these weights using a transformation $\mathbf{F}$, introducing a weight perturbation $\delta\mathbf{w} = \mathbf{w}' - \mathbf{w}$. Now consider a weight decoding mechanism with transformation $\mathbf{G}$, where $\mathbf{G}$ approximates the inverse of $\mathbf{F}$. Then, the weight perturbation error after decoding the weights can be bounded as follows:*

$$\|\mathbf{G}(\mathbf{F}(\mathbf{w})) - \mathbf{w}\| \leq \|\delta\mathbf{w}\|.$$

The above proposition demonstrates the potential of a weight decoding strategy to reduce weight perturbation and minimize the impact on model performance at inference. Inspired by Proposition 3.2, we aim to design a novel Encoder-Decoder-based model compression approach that can enable the potential of achieving a high compression ratio while minimizing inference loss.

4. Proposed Method: DeepCompress-ViT

We propose DeepCompress-ViT, a novel compression framework designed for achieving a high compression ratio while preserving inference accuracy. As shown in Figure 3, our novel DeepCompress-ViT framework consists of two stages. In the first stage, we propose *Unified Compression Training*, which jointly trains the encoder and decoder networks (see Figure 3). The encoder network, denoted as $\mathbf{F}$, learns to encode the original model weights $\mathbf{w}$ from $\mathbb{R}^d$ to $\mathbb{R}^r$, where $r \ll d$. And the decoder network, $\mathbf{G}$, learns to decode the encoded weights back to $\mathbb{R}^d$. After training is complete, only the encoded weights and the decoder network are retained for inference, while the encoder network is discarded. During inference, the proposed DeepCompress-ViT reconstructs each layer's weight one by

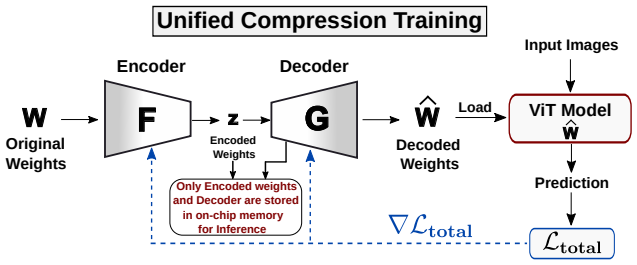

Figure 3. *Unified Compression Training. The Encoder and Decoder parameters are optimized jointly to minimize the loss. After training, only encoded weights and decoder network is retained for inference*

one and performs inference using the decoded version of the weight to help maintain inference accuracy.

While the decoding step is necessary to maintain accuracy, it has two drawbacks. To understand this, lets consider a simplified form the output computation for a given layer: $input \times encoded\ weight \times decoder\ weight$. First, the decoding step will be performed using $encoded\ weight \times decoder\ weight$ which introduces additional computation at runtime, and second, the reconstructed version of the weight will increase temporary on-chip memory consumption, especially for larger layers. To address this, we propose optimizing the decoding step by designing a lightweight decoder implemented as a single linear layer. In addition, we leverage the associative property of matrix multiplication to transform the inefficient weight decoding $input \times (encoded\ weight \times decoder\ weight)$ to optimized test-time decoding as $(input \times encoded\ weight) \times decoder\ weight$, which would ensure the computation cost remains unchanged for an optimum level of decoder input (compressed weight size r outlined in equation 4 in details) is selected. This optimization ensures the proposed DeepCompress-ViT does not introduce any computational overhead (Figure 4) in pursuit of compression and accuracy.

4.1. Unified Compression Training (UCT)

In this stage, an encoder-decoder network is trained to achieve aggressive compression. The encoder network $\mathbf{F}$, learns to encode the model's weights to a compressed representation as the encoded embedding, while the decoder network, $\mathbf{G}$, learns to decode the weights back to their original dimension as shown in Figure 3.

Consider a ViT model $\mathbf{H_W}(\cdot)$ consisting of L layers, where the weights of the layers are denoted by the set $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_L\}$. Let the parameters of the encoder network $\mathbf{F}(.)$ be represented as $\mathbf{W_E}$, and the parameters of the decoder network $\mathbf{G}(.)$ be represented as $\mathbf{W_D}$.

During training, each layer's weight $\mathbf{w}_i$ is passed through the encoder network $\mathbf{F}$, which encodes it to a lower-dimensional representation, given by $\mathbf{z}_i = \mathbf{F}(\mathbf{w}_i)$. The decoder network $\mathbf{G}$ then decodes $\mathbf{z}_i$ back to its original dimensionality as $\hat{\mathbf{w}}_i = \mathbf{G}(\mathbf{z}_i)$. The encoder and decoder networks are trained jointly to minimize the mean squared error (MSE) between the original weight set ($\mathbf{W}$) and the reconstructed weight set ($\hat{\mathbf{W}}$). The weight reconstruction loss is calculated as:

$$\mathcal{L}_{\mathbf{mse}} = \frac{1}{L}\sum_{i=1}^{L} \|\mathbf{w}_i - \mathbf{G}(\mathbf{F}(\mathbf{w}_i))\|^2$$

To optimize this training step further, we unify the encoder-decoder training step with the original ViT-model. Our hypothesis is that only MSE loss-based weight reconstruction may not provide optimal performance [2]. Even though minimizing MSE loss results in the minimization of weight perturbation error, different sets of decoded weights with the same MSE value can have varying weight distributions, which can lead to a performance drop in the ViT model. Hence, it is reasonable to incorporate the ViT model into this training loop to align the ViT model's weight distribution with the decoded version of the weight.

Hence, we propose this end-to-end unified framework where the encoder-decoder networks are jointly trained with the ViT model. Our method will incorporate cross-entropy (CE) loss and knowledge distillation (KD) from a pre-trained model in training, where CE ensures the ViT model performance is optimized on the decoded weight values w.r.t ground truth, and KD ensures the decoded weight parameters do not drift far apart from the initial weight distribution of a pre-trained ViT model.

The forward pass for the whole end-to-end training after obtaining the decoded weight sets $\hat{\mathbf{W}}$ from the decoder is formulated as $\mathbf{H_W}(\cdot) \rightarrow \mathbf{H_{\hat{W}}}(\cdot)$, which implies that the ViT model inference will always use the decoded weights, i.e., the forward pass of the ViT model becomes $\hat{\mathbf{y}} = \mathbf{H_{\hat{W}}}(\mathbf{x})$, where $\mathbf{x}$ are the input images and $\hat{\mathbf{y}}$ are the predictions from the ViT model. Next, using $\hat{\mathbf{y}}$, we compute the cross-entropy loss and knowledge distillation loss as follows:

$$\mathcal{L}_{\mathbf{CE}} = -\sum_i \mathbf{y}_i \log(\hat{\mathbf{y}}_i), \quad \mathcal{L}_{\mathbf{KD}} = \sum_i \hat{\mathbf{y}}_{\mathrm{t},i} \log\left(\frac{\hat{\mathbf{y}}_{\mathrm{t},i}}{\hat{\mathbf{y}}_i}\right)$$

where $\hat{\mathbf{y}}_{\mathrm{t}}$ represents the prediction from the pretrained teacher model. The total loss function is then defined as:

$$\mathcal{L}_{\mathbf{total}} = \mathcal{L}_{\mathbf{mse}} + \alpha \cdot \mathcal{L}_{\mathbf{CE}} + \beta \cdot \mathcal{L}_{\mathbf{KD}},$$

where α and β are scaling factors controlling the contributions of cross-entropy and knowledge distillation losses, respectively. Finally, the overall loss function is minimized by jointly optimizing the encoder and decoder parameters $\mathbf{W_E}$ and $\mathbf{W_D}$ as follows:

$$\min_{\mathbf{W_E}, \mathbf{W_D}} \mathcal{L}_{\mathbf{total}}$$

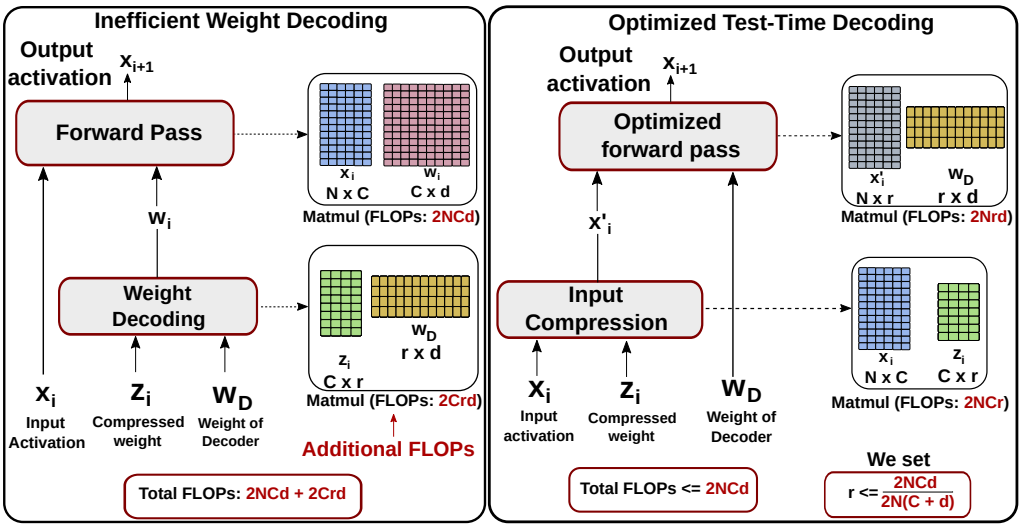

Figure 4. *Proposed Optimized Test-Time Decoding overview. Decoding the encoded (compressed) weights first and then matrix multiplying with input activation increases FLOPs. The proposed method eliminates extra FLOPs by multiplying input activation with encoded weights first, then passing through decoder. The chosen value of encoding dimension (r) prevents any FLOP increase.*

After training is complete, for inference, we keep only the encoded weight parameters $\{\mathbf{z}_i\}$ and a lightweight decoder $\mathbf{W_D}$ in memory while discarding the encoder $\mathbf{W_E}$.

At inference, the decoder needs to map compressed weights to their original form layer by layer, which may help preserve accuracy. However, this can introduce two major drawbacks. First, the decoding step will inevitably lead to additional computation at runtime, and second, the decoded weights will increase temporary on-chip memory consumption, particularly for larger layers. Next, we outline our optimized test-time decoding strategy, which ensures these two bottlenecks remain in check.

4.2. Optimized Test-Time Decoding

To mitigate the overhead associated with the decoding step, first design level optimization we propose is to design the decoder $\mathbf{G}$ as a single linear layer. This design choice minimizes repetitive decoding operations, reducing the computational overhead. Although a single-layer decoder network lowers the computational load relative to a multi-layer decoder, the computation needed for decoding weights using a single layer still persists.

To illustrate the overhead of weight decoding step, consider a fully connected layer in a ViT model with original weights $\mathbf{w}_i \in \mathbb{R}^{C\times d}$, whose weights are encoded and the encoded weights are represented by $\mathbf{z}_i \in \mathbb{R}^{C\times r}$. During inference of this layer, the weights are first decoded using the single-layer decoder network $\mathbf{G}$, where weights of the decoder network is represented by $\mathbf{w}_D \in \mathbb{R}^{r\times d}$. This weight decoding computation can be represented as follows:

$$\hat{\mathbf{w}}_i = \mathbf{z}_i \times \mathbf{w}_D, \tag{1}$$

where $\hat{\mathbf{w}}_i \in \mathbb{R}^{C\times d}$ denotes the decoded weights. One of the most widely used metrics to evaluate computational complexity is Floating-point Operations (FLOPs) [6], which counts the total number of multiplications and additions performed in a given operation. The FLOPs incurred for the weight decoding operation is $2 \cdot C \cdot r \cdot d$, corresponding to the matrix multiplication in eqn. (1).

Next, the usual forward pass of fully connected layer in the ViT model with input activations $\mathbf{x}_i \in \mathbb{R}^{N\times C}$ and decoded weights $\hat{\mathbf{w}}_i$ is computed as follows:

$$\mathbf{x}_{i+1} = \mathbf{x}_i \times \hat{\mathbf{w}}_i, \tag{2}$$

where $\mathbf{x}_{i+1}$ represents the output activations of the layer. The FLOPs incurred by this forward pass is $2 \cdot N \cdot C \cdot d$, resulting from the matrix multiplication in eqn. (2). Therefore, the total FLOPs incurred in the forward pass with this strategy is $(2 \cdot N \cdot C \cdot d + 2 \cdot C \cdot r \cdot d)$ where the additional $2 \cdot C \cdot r \cdot d$ FLOPs is incurred because of this inefficient weight decoding strategy.

Next, we optimize the decoding step to eliminate this additional $2 \cdot C \cdot r \cdot d$ FLOPs, we propose *Optimized Test-Time Decoding*. Our proposed decoding approach leverages the associative property of matrix multiplication to reorder computation sequence to maintain or improve overall FLOPs. To illustrate, recall that the overall computation for inefficient decoding strategy is as follows:

$$\mathbf{x}_{i+1} = \mathbf{x}_i \times \hat{\mathbf{w}}_i = \mathbf{x}_i \times (\mathbf{z}_i \times \mathbf{w}_D), \tag{3}$$

where it shows weight decoding first and then perform the forward pass in the fully connected layer of ViT model. We optimize the decoding step by simply reorder the matrix multiplications as follows:

$$\mathbf{x}_{i+1} = \mathbf{x}'_i \times \mathbf{w}_D = (\mathbf{x}_i \times \mathbf{z}_i) \times \mathbf{w}_D,$$

where we first perform input compression ($\mathbf{x} \in \mathbb{R}^{N \times C} \rightarrow \mathbf{x}' \in \mathbb{R}^{N \times r}$) and then do the optimized forward pass ($\mathbf{x}' \times \mathbf{w}_D$). The proposed strategy is also illustrated in Figure 4.

The total FLOPs for this optimized approach is $2 \cdot N \cdot r \cdot (C + d)$. However, with an arbitrary value of r, the overall FLOPs can still go up. Therefore, to ensure that the total FLOPs of the proposed decoding strategy do not exceed the baseline models FLOPs, we derive the optimal r for efficiency by solving the following inequality,

$$2 \cdot N \cdot r \cdot (C + d) \leq 2 \cdot N \cdot C \cdot d,$$

where $2 \cdot N \cdot C \cdot d$ is the FLOPs incurred due to normal forward pass in eqn. (2). Solving the inequality, we get:

$$r \leq \frac{2 \cdot N \cdot C \cdot d}{2 \cdot N \cdot (C + d)} \tag{4}$$

In summary, by reordering the computation sequence and using optimal r for compression derived in eqn. (4), our innovative decoding optimization does not incur any additional computation.

5. Experimental Results

5.1. Datasets and Implementation Details

Datasets. We conduct our experiments on the ILSVRC12 ImageNet classification dataset [22] consisting 1,000 classes with 1.2 million training images and 50k validation images. Additionally, we evaluate DeepCompress-ViT on the widely used COCO benchmark dataset [14] for object detection and instance segmentation task.

Encoder-Decoder Architecture. We use four encoder network each with 2 linear layers to encode four types of layers: `attn_qkv`, `attn_proj`, `mlp_fc1` and `mlp_fc2`. Each encoder has an affine transformation before and after the 2 linear layers. Similarly to encoders, we use four *single* linear layer decoders to decode the four types of layers. Note that each decoder is a shared decoder which decodes a particular type of layer of *all* ViT blocks. Note that, we exclude the first layer, last layer, and LayerNorm parameters from compression.

Experimental Settings. The encoder and decoder models are trained for 275 epochs with a batch size of 256 and a base learning rate of 1×10^{-4}, using a cosine learning rate scheduler. For data augmentation, we apply the standard techniques used in [27] and for knowledge distillation, we use pretrained distilled DeiT models [27] with temperature $\tau = 1$. For UCT, we set the value of $\alpha = 1$ and $\beta = 3 \times 10^3$. The encoded weights are quantized using 4-bit asymmetric quantization, while decoder weights are quantized using 4-bit symmetric quantization, both trained using Quantization Aware Training. Additionally, the patch embedding (first) and classification (last) layers are quantized using 8-bit symmetric quantization. The optimal value of r from solving (4) is set to 277 for DeiT-S and 502 for DeiT-B. For object detection and instance segmentation, we use the mmdetection library with the standard $1\times$ training scheduler and adopt an edge-friendly resolution of 384×384 for both training and evaluation.

5.2. Results

Image Classification. Performance of the compressed DeiT models with proposed DeepCompress-ViT on the classification task is shown in Table 3. The results demonstrates substantial reductions in model size and energy consumption resulting from DeepCompress-ViT. For instance, the DeepCompress-DeiT-S model achieves a compression ratio of *14.9*$\times$, reducing the model size from 84.1 MB to just 5.64 MB while maintaining an accuracy of 78.74% on ImageNet. Similarly, DeepCompress-DeiT-B achieves a compression ratio of *17.4*$\times$, decreasing the model size from 330.2 MB to 18.9 MB. Remarkably, both models show an *unprecedented energy reduction of 673*$\times$ *and 1475*$\times$ *and latency reduction of 44.5*$\times$ *and 68.4*$\times$ respectively (please refer to section 5.3 and Figure 5 for more hardware energy and latency evaluation details). Proposed DeepCompress-ViT achieves this unprecedented level of model compression, energy, and latency reduction, enabling the deployment of ViT models at the edge while only sacrificing a marginal level of accuracy (e.g., for DeiT-B only $\sim 0.1\%$)

Object Detection and Instance Segmentation. Next, table 4 compares the performance of DeepCompress-ViT with baseline (no compression) DeiT-S model on object detection and instance segmentation tasks. We trained both the models on COCO dataset integrating the respective ViT backbones with Mask R-CNN [9]. We evaluate both of the models for these tasks at an edge-friendly resolution of 384×384. Notably, DeepCompress-ViT reduces the ViT backbone (DeiT-S) size from 82.65MB to 5.27MB (15.7$\times$ compression) with a slight drop in AP^b and AP^m, further highlighting the effectiveness of our compression method on other complex vision tasks.

Comparison with SOTA compression methods. Finally, Table 5 provides a comparative analysis of DeiT-Small models compressed using different pruning and quantization methods. Among the methods, DeepCompress-ViT achieves the highest compression ratio of $14.9\times$, reducing the model size to 5.64 MB while maintaining a top-1 accuracy of 78.74%. In contrast, popular WDPruning [31], LSQ [8] and Q-ViT [11] achieves achieve much lower accuracies of 60.51%, 68.00% and 71.9%, respectively. Our proposed DeepCompress-ViT consistently outperforms them by large margins of $\sim$ 18 %, $\sim$ 10 %, and $\sim$ 7%, respectively, making DeepCompress-ViT the only practical choice to maintain accuracy while performing aggressive compression.

Table 3. ***Performance of DeepCompress-ViT on ImageNet-1K***. *DeepCompress-ViT achieves memory* ($\geq 15\times$) *and energy consumption* ($\geq 1531\times$), *while marginal accuracy drop (<1 %).*

Method	Accuracy (%)	Model Size (MB)	Compression Ratio	GFLOPs	Execution Time (s)	Energy Consumption (J)
DeiT-S [27]	79.72	84.1	1x	9.20	24.02	9979.52
DeepCompress-DeiT-S	78.74	5.64	14.9x	8.88	0.54	14.83
DeiT-B [27]	81.74	330.2	1x	35.14	129.32	150895
DeepCompress-DeiT-B	81.62	18.9	17.4x	30.84	1.89	102.29

Table 4. *Performance of DeepCompress-ViT for Object Detection on the COCO validation set with Mask R-CNN evaluated at an edge-friendly resolution of 384×384.*

Method	AP^b	AP^m	ViT Backbone Size (MB)
DeiT-S [27]	26.2	23.7	82.65
DeepCompress-ViT	23.0	20.9	5.27 (15.7×)

Table 5. *Performance comparison of DeiT-Small with SOTA compression methods on ImageNet-1K*

Model	Accuracy (%)	Model Size (MB)	Compression Ratio
DeiT-S [27]	79.72 (0)	84.1	1x
WDPruning [31]	60.51 (↓19.21)	33.7	2.5x
LSQ [8]	68.00 (↓11.72)	6.18	13.6x
Q-ViT [11]	71.90 (↓7.82)	6.18	13.6x
DeepCompress-ViT	78.74 (↓0.98)	5.64	14.9x

5.3. Hardware Evaluation

High compression gain gives an understanding of the effectiveness of a compression method in terms of memory savings. But to successfully realize the full potential of our proposed method, we evaluate the energy and latency gain on hardware evaluation platforms. For hardware deployment, we use the gem5-Aladdin platform [25], which effectively captures dynamic interactions between accelerators and the SoC platform, enabling detailed architectural analysis of the proposed compression method. Our setup incorporates a cycle-accurate in-house-developed model based on the Eyeriss accelerator [5] (0.0029 DRAM access/multiply and accumulation (MAC), 33.6 GMAC/s). Performance metrics from gem5-Aladdin are then analyzed using DRAMSim2 [21] to model realistic off-chip memory latency and energy characteristics. The total on-chip weight memory is capped at three configurations (2 MB, 4 MB, 8 MB). The analyzed DeiT-S model initially requires 84.1MB of memory, which exceeds the on-chip memory capacity. By applying DeepCompress-ViT algorithm, this requirement is reduced to 5.64 MB, which can readily fit into the 8MB on-chip memory. However, with a smaller memory capacity, it is not feasible to transfer the entire set of weight parameters to the accelerator in a single batch due to the model's memory demands. Therefore, we implemented a weight parser that transmits weight parameters in variable batch sizes to the accelerator. Figure 5 shows the breakdown of energy consumption and latency across MAC, off-chip, and on-chip memory access components for various on-chip memory sizes. For both DeiT-S and DeiT-B, the DeepCompress-ViT algorithm achieves significant energy savings, *reducing consumption by approximately 673× and 1475×*, respectively, with an 8MB on-chip memory. This remarkable reduction is primarily due to the DeepCompress-ViT algorithm's ability to minimize off-chip memory access, as visible through the high difference in the red bar plots. This reduction becomes even more pronounced with smaller memory sizes, enabling for the first time the possibility of deploying large ViT models at the edge and performing time-sensitive applications.

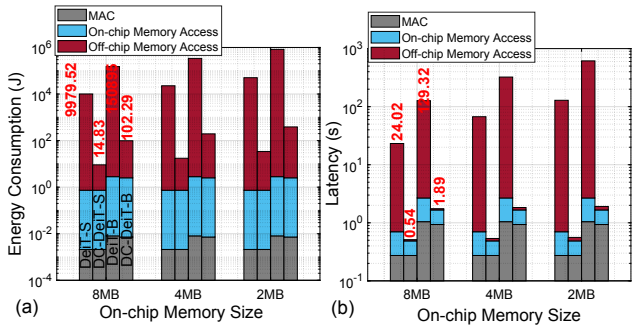

Figure 5. *Breakdown of (a) Energy consumption, (b) Latency vs. on-chip memory size. For every off-chip memory size, DeiT-S, DeepCompress-ViT-S, DeiT-B, and DeepCompress-ViT-B results are plotted, respectively.*

6. Conclusion

In this paper, we present DeepCompress-ViT, a novel approach to achieve highly compressed Vision Transformers optimized for efficient edge deployment. We first analyze the limitations of existing model compression techniques and show that they lead to large weight perturbation error because of their lossy encoding mechanism. Then, to resolve this challenge, we develop an orthogonal compression technique that consists of an encoding for compression and a decoding step to help maintain performance. However, decoding weights repeatedly during inference can lead to additional computation and increase on-chip memory usage, both of which are critical concerns for edge applications. To address these challenges, we propose a novel optimized Test-Time Decoding scheme that enables efficient inference without increased computation and avoids high on-chip memory usage. Our extensive experiments and hardware evaluation show that the proposed DeepCompress-ViT can lead to an unprecedented reduction in energy and execution time with minimal loss of performance, unlocking the possibilities of deploying large-scale ViT models at edge IIoT devices.

References

[1] Esp32-s3 designed for aiot applications, 2022. 2

[2] Sabbir Ahmed, Uday Kamal, and Md Kamrul Hasan. Dfr-tsd: A deep learning based framework for robust traffic sign detection under challenging weather conditions. *IEEE Transactions on Intelligent Transportation Systems*, 23(6): 5150–5162, 2021. 5

[3] Jiang Bian, Abdullah Al Arafat, Haoyi Xiong, Jing Li, Li Li, Hongyang Chen, Jun Wang, Dejing Dou, and Zhishan Guo. Machine learning in real-time internet of things (iot) systems: A survey. *IEEE Internet of Things Journal*, 9(11): 8364–8386, 2022. 1

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 1

[5] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE journal of solid-state circuits*, 52(1):127–138, 2016. 8

[6] Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. Compute and energy consumption trends in deep learning inference. *arXiv preprint arXiv:2109.05472*, 2021. 6

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 3

[8] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019. 1, 3, 7, 8

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 7

[10] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, pages 10–14. IEEE, 2014. 2, 3

[11] Yanjing Li, Sheng Xu, Baochang Zhang, Xianbin Cao, Peng Gao, and Guodong Guo. Q-vit: Accurate and fully quantized low-bit vision transformer. *Advances in neural information processing systems*, 35:34451–34463, 2022. 1, 2, 3, 4, 7, 8

[12] Zhikai Li, Liping Ma, Mengjuan Chen, Junrui Xiao, and Qingyi Gu. Patch similarity aware data-free quantization for vision transformers. In *European conference on computer vision*, pages 154–170. Springer, 2022.

[13] Zhikai Li, Junrui Xiao, Lianwei Yang, and Qingyi Gu. Repq-vit: Scale reparameterization for post-training quantization of vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17227–17236, 2023. 3

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014. 7

[15] Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. Fq-vit: Post-training quantization for fully quantized vision transformer. *arXiv preprint arXiv:2111.13824*, 2021. 3, 4

[16] Ji Liu, Dehua Tang, Yuanxian Huang, Li Zhang, Xiaocheng Zeng, Dong Li, Mingjie Lu, Jinzhang Peng, Yu Wang, Fan Jiang, et al. Updp: A unified progressive depth pruner for cnn and vision transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13891–13899, 2024. 3

[17] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 3

[18] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. *Advances in Neural Information Processing Systems*, 34:28092–28103, 2021. 3, 4

[19] Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024. 4

[20] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 1

[21] Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. Dramsim2: A cycle accurate memory system simulator. *IEEE computer architecture letters*, 10(1):16–19, 2011. 8

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1, 7

[23] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 1

[24] Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in neural information processing systems*, 33:20378–20389, 2020. 3

[25] Yakun Sophia Shao, Sam Likun Xi, Vijayalakshmi Srinivasan, Gu-Yeon Wei, and David Brooks. Co-designing accelerators and soc interfaces using gem5-aladdin. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12. IEEE, 2016. 8

[26] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272, 2021. 1

[27] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 1, 3, 7, 8

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 3

[29] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 568–578, 2021. 3

[30] Huanrui Yang, Hongxu Yin, Maying Shen, Pavlo Molchanov, Hai Li, and Jan Kautz. Global vision transformer pruning with hessian-aware saliency. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18547–18557, 2023. 3

[31] Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3143–3151, 2022. 1, 2, 3, 4, 7, 8

[32] Lu Yu and Wei Xiang. X-pruner: explainable pruning for vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24355–24363, 2023.

[33] Shixing Yu, Tianlong Chen, Jiayi Shen, Huan Yuan, Jianchao Tan, Sen Yang, Ji Liu, and Zhangyang Wang. Unified visual transformer compression. *arXiv preprint arXiv:2203.08243*, 2022. 3

[34] Zhengqing Yuan, Rong Zhou, Hongyi Wang, Lifang He, Yanfang Ye, and Lichao Sun. Vit-1.58 b: Mobile vision transformers in the 1-bit era. *arXiv preprint arXiv:2406.18051*, 2024. 4

[35] Jiale Zhang, Bing Chen, Yanchao Zhao, Xiang Cheng, and Feng Hu. Data security and privacy-preserving in edge computing paradigm: Survey and open issues. *IEEE access*, 6: 18209–18237, 2018. 1

[36] Chuanyang Zheng, Kai Zhang, Zhi Yang, Wenming Tan, Jun Xiao, Ye Ren, Shiliang Pu, et al. Savit: Structure-aware vision transformer pruning via collaborative optimization. *Advances in Neural Information Processing Systems*, 35:9010–9023, 2022. 3