HaLViT: Half of the Weights are Enough

Onur Can Koyun Behçet Uğur Töreyin
Informatics Institute
Signal Processing for Computational Intelligence Research Group (SP4CING)
Dept. of Artificial Intelligence and Data Engineering
İstanbul Technical University, İstanbul, Türkiye
{okoyun, toreyin}@itu.edu.tr

Abstract

Deep learning architectures like Transformers and Convolutional Neural Networks (CNNs) have led to groundbreaking advances across numerous fields. However, their extensive need for parameters poses challenges for implementation in environments with limited resources. In our research, we propose a strategy that focuses on the utilization of the column and row spaces of weight matrices, significantly reducing the number of required model parameters without substantially affecting performance. This technique is applied to both Bottleneck and Attention layers, achieving a notable reduction in parameters with minimal impact on model efficacy. Our proposed model, HaLViT, exemplifies a parameter-efficient Vision Transformer. Through rigorous experiments on the ImageNet dataset and COCO dataset, HaLViT's performance validates the effectiveness of our method, offering results comparable to those of conventional models.

1. Introduction

Deep learning models, particularly Transformers and Convolutional Neural Networks (CNNs), have emerged as powerful tools in the field of artificial intelligence, catalyzing advancements in various domains such as computer vision, natural language processing, and speech recognition [24, 38]. However, these models often require a significant number of parameters, leading to challenges in deploying them in resource-constrained environments [2, 35].

CNNs, since their resurgence with AlexNet in 2012, have been the cornerstone of computer vision tasks, offering state-of-the-art performance in areas like image classification, object detection, and more [16, 24]. The evolution of CNN architectures, from AlexNet to more complex designs like ResNet and EfficientNet, has consistently sought to balance performance with computational efficiency [16, 35].

On the other hand, Transformers, introduced by Vaswani et al. [38], have revolutionized the field of natural language processing and more recently have made significant inroads into computer vision [9]. The Transformer's attention mechanism allows models to weigh the importance of different parts of the input data, a feature that has proven highly effective in various tasks. However, this capability comes at the cost of increased model complexity and a higher number of parameters. The evolution of Vision Transformers (ViTs) has led to a diverse range of models that further expand their applicability in computer vision. Notable among these is the original Vision Transformer (ViT), which applies the Transformer architecture directly to image patches for classification [9]. Following ViT, the Swin Transformer introduces a hierarchical structure using shifted windows, enhancing performance in tasks like object detection and semantic segmentation [29].

The DETR model integrates Transformers for end-to-end object detection, showcasing the flexibility of Transformer models in adapting to different tasks [3]. Additionally, models like DeiT and DINO have contributed to the field by focusing on data-efficient training and self-supervised learning, respectively [4, 36]. Other significant contributions include Deformable DETR for more flexible object detection [52], CCT for compactness in Transformer models [15], and NesT which proposes a nested hierarchical approach [50].

Moreover, models like PVT have been developed to provide dense predictions without relying on convolutions, indicating the potential of Transformers to replace traditional CNNs in certain applications [40]. The T2T-ViT model takes a unique approach by converting tokens to tokens, which is a novel way of training Vision Transformers from scratch [47]. Additionally, the introduction of convolutions to Vision Transformers in the CvT model demonstrates the ongoing integration and evolution of Transformer and CNN architectures [43].

The challenge of deploying these large models in

resource-limited settings, such as mobile devices or edge computing platforms, has spurred research into parameter-efficient architectures [19, 33]. Techniques like network pruning, quantization, and knowledge distillation have been explored to reduce the size and computational requirements of these models without significantly compromising their performance [14, 22, 32].

In this work, we introduce an innovative method aimed at enhancing parameter efficiency within deep learning architectures, with a particular emphasis on Transformers and Convolutional Neural Networks (CNNs). Leveraging the column and row spaces of weight matrices, we propose the HaLViT, a novel architecture that significantly reduces the parameter count of Vision Transformers (ViTs) by half. This reduction is meticulously designed to maintain performance fidelity, a claim substantiated through rigorous evaluation on the ImageNet-1k dataset. Further empirical investigations extend to the realm of transfer learning on smaller datasets, alongside object detection tasks utilizing the COCO dataset in conjunction with the Mask-RCNN framework. The primary contribution of our approach lies in its ability to offer a parameter-efficient solution for deep learning models, particularly beneficial in environments constrained by limited memory resources. The broader implications of our research transcend academic curiosity, providing a viable pathway for the deployment of artificial intelligence models in practical applications where resources are scarce.

2. Related Work

2.1. Efficient Deep Learning Models

Efficient deep learning models aim to reduce computational complexity, memory requirements, and energy consumption, making them suitable for deployment in resource-constrained environments. The principal strategies include network pruning, knowledge distillation, and low-rank factorization of weight matrices.

Network Pruning: Building on the foundational work by Han et al. [14], which demonstrated a significant reduction in neural network parameters with minimal accuracy loss, several studies have furthered the field of neural network efficiency and compression. The "deep compression" technique introduced by Han et al. [14], combines pruning, quantization, and Huffman coding to reduce neural network storage requirements effectively. This method involves pruning redundant connections, quantizing weights, and applying Huffman coding, substantially decreasing the storage and computational needs without sacrificing accuracy. Subsequent research has further refined these techniques. Magnitude-based weight pruning methods [27] have proven effective in compressing networks, and complementary strategies like quantization and low-rank matrix factorization [27] have been used alongside pruning for maximal compression. *Knowledge Distillation:* Knowledge distillation, first introduced by Hinton et al. [18], is a method for training compact and efficient models while maintaining performance comparable to larger, more computationally demanding models. This process involves transferring knowledge from a large, complex 'teacher' model to a smaller, more efficient 'student' model. The student model is trained not only to accurately predict labels but also to emulate the label distribution produced by the teacher model. This approach allows the student model to achieve performance similar to the teacher model while being more suitable for real-time applications and devices with limited computing power.

Low-Rank Factorization: This method decomposes weight matrices into lower rank approximations, reducing the number of parameters. Jaderberg et al. (2014) effectively applied this to CNNs for speed improvements [21].

2.2. Transformers

Transformers, initially introduced by Vaswani et al. (2017) for natural language processing tasks, have been adapted for various applications including image recognition [38]. The self-attention mechanism allows them to model long-range dependencies efficiently. However, Transformers are often parameter-intensive, which can be a drawback for deployment in memory-constrained environments.

Efficient Transformers: A major challenge with Transformers is their high computational and memory costs, particularly in scenarios with long sequences or large inputs [28]. This has catalyzed a wave of research dedicated to enhancing the efficiency of Transformers. For instance, numerous variants such as Reformer, Linformer [39], and Performer have been proposed, targeting improvements in computational and memory efficiency.

A key innovation in efficient Transformer design is the development of models that address the quadratic complexity of self-attention, a central feature of the Transformer architecture. By focusing on both memory and computational efficiency, these models are more suitable for applications with constrained computational resources [28]. In the context of natural language processing, the Primer model represents a significant advancement, achieving higher efficiency in auto-regressive language modeling through architectural changes such as squaring ReLU activations and adding depthwise convolution layers [34].

The application of efficient Transformers extends beyond language tasks to fields such as computer vision. The SegFormer [44], for instance, integrates a novel Transformer encoder with a lightweight MLP decoder for semantic segmentation, demonstrating significant improvements in model size, runtime, and accuracy on standard datasets like ADE20K[51] and Cityscapes [7]. This approach not

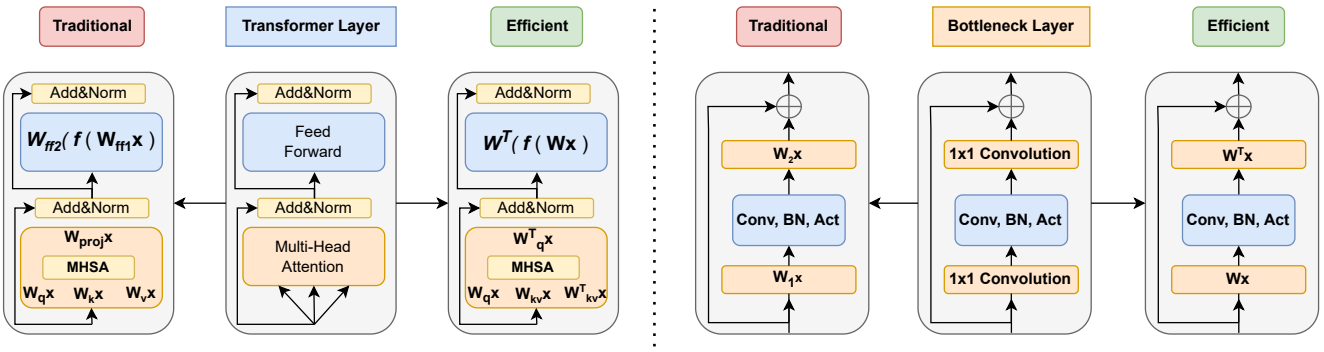

Figure 1. An overview of proposed method in Transformer and Bottleneck layers. Our method employs a single weight matrix, $\mathbf{W}$, across the Multi-Head Self-Attention (MHSA), feed-forward, and bottleneck layers, resulting in parameter-efficient configurations compared to conventional methods.

only consolidates the Transformer's robustness and accuracy but also ensures efficiency, a critical aspect for real-world deployment.

2.3. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks are predominantly used in image processing and computer vision tasks. Their layered structure allows for effective feature extraction and pattern recognition in images. LeCun et al. (1998) were pioneers in demonstrating the effectiveness of CNNs in digit recognition [25]. CNNs have experienced considerable development from their early days, marked by substantial advancements in different areas. Architectural innovations have been pivotal, with the creation of AlexNet [24], MobileNet [19], ResNet [16], EfficientNet [35], and GhostNet [12], each representing significant breakthroughs in network design and operational efficiency.

Parameter Efficiency in CNNs: The recent advancements in Convolutional Neural Networks (CNNs), particularly in terms of parameter efficiency, have been significantly influenced by developments in depthwise separable convolutions. This approach, exemplified in MobileNets [19], substantially reduces the number of parameters while maintaining competitive performance.

Extending this notion, EfficientNet [35] represents a groundbreaking stride in CNN design. It employs a compound scaling method that uniformly scales network dimensions (depth, width, and resolution) using a fixed coefficient, leading to considerable gains in efficiency and performance. EfficientNet models, especially EfficientNet-B7, have achieved state-of-the-art results on benchmarks like ImageNet, CIFAR-100, and Flowers, surpassing previous models in accuracy while being significantly smaller and faster. The EfficientNet architecture, starting from the baseline model EfficientNet-B0, leverages a multi-objective neural architecture search optimizing both accuracy and computational cost (FLOPS), leading to a network with fewer parameters and higher efficiency [35]. This methodical scaling and optimization yield models that are not only compact but also computationally less demanding, allowing for higher performance with reduced resource usage.

3. Method

In this paper, our goal is to demonstrate that leveraging the row space and column space of weight matrix $\mathbf{W}$ can yield comparable performance with conventional approach. By conducting a series of experiments and analyses, we intend to showcase how these vector spaces contribute to the parameter efficiency of deep learning models. Using this strategy, we introduce HaLViT a parameter-efficient variant of the Vision Transformer.

Let $\mathbf{W}$ be an $m \times n$ matrix in a deep neural network layer, and $\mathbf{x}$ be a feature vector. The operation of $\mathbf{W}$ on $\mathbf{x}$ results in a linear combination of $\mathbf{W}$'s columns, where the components of $\mathbf{x}$ serve as the coefficients. This can also be interpreted as projecting $\mathbf{x}$ into the row space of $\mathbf{W}$. Therefore, the output $\mathbf{y} = \mathbf{W}\mathbf{x}$ inevitably resides in the column space of $\mathbf{W}$. However, the presence of nonlinear activation functions in deep neural networks alters this characteristic. With the application of a nonlinear function $\mathcal{F}(\cdot)$, the resulting $\mathbf{y} = \mathcal{F}(\mathbf{W}\mathbf{x})$ no longer confines itself to the column space of $\mathbf{W}$. This change occurs as the nonlinear transformation of $\mathbf{W}\mathbf{x}$ shifts its dimensional orientation, moving it beyond the scope of $\mathbf{W}$'s column space. In that case, $\mathbf{W^T}\mathbf{y}$ does not reside in the row space of matrix $\mathbf{W}$, implying that both the row and column spaces of matrix $\mathbf{W}$ can be independently utilized in each layer to reduce the number of parameters.

If one considers the matrix as a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^m$ $\mathbf{y} = \mathbf{W}\mathbf{x}$, then the column space of the matrix $\mathbf{W}$ equals the image of this linear transformation, similarly a linear transformation from $\mathbb{R}^m$ to $\mathbb{R}^n$ $\mathbf{W^T}\mathbf{y}$,

then the row space of the matrix $\mathbf{W}$ equals the image of this transformation. Introducing nonlinearity between these projections results in a nonlinear transformation akin to that found in traditional deep neural network layers: $\mathbf{W^T}\mathcal{F}(\mathbf{Wx})$.

In our approach, we have applied this concept to Transformer encoder layers in Vision Transformers (ViTs) and bottleneck layers in Residual Networks (ResNets), with a focus on reducing parameter count. Particularly, within the Transformer encoder layers of ViTs, our approach effectively reduces the parameter count by half.

3.1. Transformer Encoder Layer

Multi-Head Attention. The standard implementation of Multi-Head Attention (MHA) in Transformer models relies on separate weight matrices $\mathbf{W_q}$, $\mathbf{W_k}$, and $\mathbf{W_v}$ to project the input feature vector $\mathbf{x}$ into queries (Q), keys (K), and values (V) spaces respectively. Additionally, after computing the attention scores and aggregating the values, a final linear projection is applied using a separate weight matrix $\mathbf{W_{proj}}$. This conventional approach, while effective, requires a substantial number of parameters, especially as the size of the model scales.

In an effort to enhance parameter efficiency without compromising the model's ability to learn complex representations, we propose an innovative approach that leverages the row and column spaces of new weight matrices $\mathbf{W_q}, \mathbf{W_q^T}$ and $\mathbf{W_{kv}}, \mathbf{W_{kv}^T}$. Specifically, we project the input feature vector $\mathbf{x}$ into the query, key, and value spaces using a shared matrix for keys and values ($\mathbf{W_{kv}}$) and a separate matrix for queries ($\mathbf{W_q}$). This significantly reduces the number of parameters. The Multi-Head Attention operation and subsequent linear projection under this proposed framework can be expressed as follows:

$$\hat{\mathbf{x}} = MHA(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = MHA(\mathbf{W_q x}, \mathbf{W_{kv} x}, \mathbf{W_{kv}^T x}), \tag{1}$$

where $\mathbf{W_q x}$ generates the queries, and both keys and values are derived from $\mathbf{W_{kv} x}$ and its transpose $\mathbf{W_{kv}^T x}$, respectively. Following the computation of attention scores and the aggregation of value vectors, the resulting vector $\hat{\mathbf{x}}$ undergoes a linear projection via:

$$Proj(\hat{\mathbf{x}}, \mathbf{W_q^T}) = \mathbf{W_q^T}\hat{\mathbf{x}}, \tag{2}$$

where $\mathbf{W_q^T}$ is used for the final projection, further emphasizing the utilization of both row and column spaces of the weight matrices.

This method essentially halves the number of parameters in the Multi-Head Attention (MHA) layer by utilizing a shared weight matrix for both keys and values and leveraging its transpose. This reduction is achieved without significantly compromising the model's performance, as preliminary results indicate. The underlying intuition is that the column and row spaces of $\mathbf{W_{kv}}$ and $\mathbf{W_q}$ can capture sufficient information for effective key-value pairing and query projection, respectively. Furthermore, this approach introduces an interesting avenue for exploring the dual roles of weight matrices in Transformer architectures, potentially leading to more parameter-efficient models without sacrificing their expressive power.

Feed Forward Network. The Feed Forward Network (FFN) within a Transformer layer traditionally consists of a layer normalization step followed by two linear feed-forward layers with an interposed nonlinear activation function. The structure of the FFN is crucial for adding non-linearity and depth to the model's capability to process sequences. Typically, these feed-forward layers are characterized by their weight matrices $\mathbf{W_1}$ and $\mathbf{W_2}$, and bias vectors $\mathbf{b_1}$ and $\mathbf{b_2}$. Given a feature vector $\mathbf{x}$ and a nonlinear activation function $\mathcal{F}(\cdot)$, the conventional formulation of an FFN's operation can be represented as follows:

$$FFN(\mathbf{x}) = \mathbf{W_2}\mathcal{F}(\mathbf{W_1 x} + \mathbf{b_1}) + \mathbf{b_2}. \tag{3}$$

This traditional approach, while effective, necessitates a significant number of parameters due to the use of separate weight matrices for each feed-forward layer. In an effort to enhance the parameter efficiency of the FFN, instead of utilizing two distinct weight matrices $\mathbf{W_1}$ and $\mathbf{W_2}$, our approach leverages the column space and row space of a single weight matrix $\mathbf{W}$. This strategy substantially reduces the total parameter count by half, without a significant compromise in the network's performance.

The redefined equation for the FFN, under this new paradigm, is as follows:

$$FFN(\mathbf{x}) = \mathbf{W^T}\mathcal{F}(\mathbf{Wx} + \mathbf{b_1}) + \mathbf{b_2}. \tag{4}$$

In this optimized formulation, $\mathbf{W}$ serves a dual purpose, linear projection of the input vector $\mathbf{x}$ in combination with the bias $\mathbf{b_1}$ and the nonlinear activation $\mathcal{F}$, and subsequently in its transposed form $\mathbf{W^T}$ to produce the final output of the FFN, with the addition of the second bias $\mathbf{b_2}$. This dual use of $\mathbf{W}$ and $\mathbf{W^T}$ not only conserves parameters but also maintains the depth and non-linearity essential for the Transformer architecture's success, presenting a promising direction for future research into more efficient neural network designs. An overview of the method is given in Figure 1.

3.2. Bottleneck Layer

Introduced in the paper "Deep Residual Learning for Image Recognition" by He et al., ResNet is one of the first architectures to use bottleneck layers effectively. Architectures in the literature, leverages bottleneck layers in different ways,

but with a common goal: to increase network efficiency by reducing the computational complexity while maintaining or improving model performance.

Let $\mathcal{G}$ be a function consists of 3×3 convolution, normalization and nonlinear activation function. The bottleneck layer can be described as follows:

$$Bottleneck(\mathbf{x}) = \mathbf{W_1^T}\mathcal{G}(\mathbf{W_2 x}), \tag{5}$$

where $\mathbf{W_1 x}$ and $\mathbf{W_2 x}$ represent 1×1 convolutions in bottleneck layer. Utilizing column and row spaces of weight matrix W, equation becomes;

$$Bottleneck(\mathbf{x}) = \mathbf{W^T}\mathcal{G}(\mathbf{W x}). \tag{6}$$

Implementing our method in the Bottleneck layer reduces the number of parameters, albeit less effectively compared to its application in the Transformer layer. Consequently, we have adopted a strategy of weight sharing in each stage in ResNets to further decrease the parameter count, without adversely affecting performance. Let $\mathcal{G}_1$, $\mathcal{G}_2$, ... ,$\mathcal{G}_n$ be functions consist of 3×3 convolution, normalization and nonlinear activation function and n is the number of bottleneck layers in a particular stage of residual network. The bottleneck layers in this stage can be written as:

$$Bottleneck(\mathbf{x_1}) = \mathbf{W^T}\mathcal{G}_1(\mathbf{W x_1}) + \mathbf{x_1}, \tag{7}$$

$$Bottleneck(\mathbf{x_2}) = \mathbf{W^T}\mathcal{G}_2(\mathbf{W x_2}) + \mathbf{x_2}, \tag{8}$$

$$Bottleneck(\mathbf{x_n}) = \mathbf{W^T}\mathcal{G}_n(\mathbf{W x_n}) + \mathbf{x_n}. \tag{9}$$

By sharing weights in same stages of the network, we effectively reuse the same set of parameters for multiple operations, thereby reducing the overall parameter footprint of the model. This is particularly beneficial in deep learning architectures like ResNets, where the depth of the network can lead to a large number of parameters, potentially causing issues like overfitting and increased computational load.

Moreover, weight sharing in bottleneck layers does not significantly compromise the learning capability of the network. It allows the model to generalize better by learning reusable patterns and features across different layers. This aspect is crucial in maintaining the performance of the network while reducing its complexity.

4. Experiment

In this section, we detail the experimental outcomes obtained by applying our proposed method to the classification, object detection and instance segmentation tasks. Results is presented in Tables 3, 4 and 5.

4.1. Image Classification

We detail the experimental outcomes obtained by applying our proposed method to the ImageNet-1k (IN1K) classification task [8]. For the implementation, the mmpretrain toolkit [6] was utilized for image classification. The IN1K dataset [8] comprises approximately 1.28 million training images and 50,000 validation images, distributed across 1,000 classes. For a balanced comparison, both Transformer-based and CNN-based models were trained using the training set, with the Top-1 accuracy being evaluated on the validation set. We follow the training recipe used by the DeiT [36] for HaLViT, and the same methodology was employed in Torchvision [31] for training ResNet50. Architecture and training details for HaLViT is given in Tables 1 and 2.

Table 1. Comparison between HaLViT, ViT and DeiT

Model	Layers	Hidden Size	MLP Size	Heads	Params
ViT-Base [5]	12	768	3072	12	86M
Deit-Base [36]	12	768	3072	12	86M
Deit-Small [36]	12	384	1536	6	22M
HaLViT-Tiny	12	384	1536	6	11M
HaLViT-Small	12	768	3072	12	43M

Table 2. Ingredients and hyper-parameters for HaLViT and ViT.

Method	ViT	HaLViT
Epochs	300	300
Batch Size	4096	1024
Optimizer	AdamW	AdamW
Learning Rate	0.003	0.0005 $\times$ batchsize
Learning Rate Decay	cosine	cosine
Weight Decay	0.3	0.05
Warmup Epochs	3.4	5
Label Smoothing	✗	0.1
Dropout	0.1	✗
Stochastic Depth	✗	0.1
Repeated Augmentation	✗	✓
Gradient Clipping	✓	✗
Rand Augment	✗	9 / 0.5
Mixup Probability	✗	0.8
Cutmix Probability	✗	1.0
Erasing Probability	✗	0.25

HaLViT. In evaluating the effectiveness of our proposed HaLViT architecture, we conducted extensive experiments on the ImageNet-1K (IN1K) validation set at a resolution of 224×224 pixels. The comparative analysis included various models, focusing on the balance between parameter efficiency and top-1 accuracy.

Models are trained for 300 epochs and 600 epochs. The learning rate was scaled according to the batch size using the formula: lrscaled $= \frac{\text{lr}}{512} \times$ batchsize, following the approach of Goyal et al. [11], but modifying the base value to 512 instead of 256 as in DeiT. Results were obtained using the AdamW optimizer, employing the same learning rates as ViT [9] but with significantly reduced weight decay: 0.05. For regularization, stochastic depth [20] was employed, which is particularly beneficial for the convergence of deep Transformers [10]. This technique was first introduced in Vision Transformer training by Wightman [42]. We also applied regularization methods like Mixup [49] and Cutmix [48], which contributed to improved performance.

Our HaLViT models, specifically HaLViT-T^1 and HaLViT-T^2, trained for 300 and 600 epochs respectively, demonstrated remarkable performance. With only 11.1M parameters, HaLViT-T^1 achieved a top-1 accuracy of 77.3%, while HaLViT-T^2 further improved the accuracy to 78.8%, showcasing the benefits of extended training. These results highlight our models' capability to deliver competitive performance with a substantially lower parameter count compared to other models such as PVTv2-B1, which achieved a top-1 accuracy of 78.7% with 14.0M parameters.

Moreover, the HaLViT-M variant, embodying our approach's scalability to larger models, attained a top-1 accuracy of 81.3% with 43.0M parameters. Although HaLViT employs the traditional ViT architecture, its performance matches or even exceeds various established models, such as PVT-M and ViT-Small/16. This demonstrates HaLViT's effectiveness in attaining high accuracy while utilizing fewer computational resources.

ResNet50. In our experimental setup, we use a resolution of 224×224 pixels for training images. The model optimization is conducted using Stochastic Gradient Descent (SGD). We set the initial learning rate to 0.1. Additionally, we apply a weight decay regularization factor of 1×10^{-4}. To adjust the learning rate smoothly during training, we use a cosine annealing schedule. This schedule is enhanced by an initial linear warm-up phase that lasts for 5 epochs. The warm-up phase plays a crucial role. It helps stabilize the learning process at the beginning. This prevents the model from converging prematurely to suboptimal minima.

Throughout the training regimen, we incorporate specific image augmentation techniques to enhance the robustness and generalizability of the model. These techniques include center cropping, where images are first resized to 248×248 pixels before a central crop of 224×224 pixels is extracted. Additionally, horizontal flip augmentations are applied to further diversify the training dataset, thus enabling the model to better generalize from the training data to unseen images.

Table 3. Top-1 accuracy comparison in IN1K validation set on 224×224 resolution. HaLViT-T^1 and HaLViT-M undergoes a training of 300 epochs, while HaLViT-T^2 extends its training to 600 epochs.

Method	#Params (M)	FLOPs (G)	Top-1 Acc. (%)
DeiT-T/16 [36]	5.7	1.3	72.2
PVT-T [40]	13.2	1.9	75.1
PVTv2-B1 [41]	14.0	2.1	78.7
HaLViT-T^1 (ours)	**11.1**	**4.6**	**77.3**
HaLViT-T^2 (ours)	**11.1**	**4.6**	**78.8**
DeiT-S/16 [36]	22.1	4.6	79.9
T2T-ViTt-14 [47]	22.0	6.1	80.7
PVT-S [40]	24.5	3.8	79.8
TNT-S [13]	23.8	5.2	81.3
SWin-T [29]	29.0	4.5	81.3
CvT-13 [43]	20.0	4.5	81.6
Twins-SVT-S [5]	24.0	2.8	81.3
FocalAtt-Tiny [46]	28.9	4.9	82.2
PVTv2-B2 [41]	25.4	3.9	82.0
PVTv2-B2-li [41]	22.6	4.0	82.1
CvT-21 [43]	32.0	7.1	82.5
T2T-ViTt-19 [47]	39.0	9.8	81.4
PVT-M [40]	44.2	6.7	81.2
PVTv2-B3 [41]	45.2	6.9	83.2
SWin-S [29]	50.0	8.7	83.0
Twins-SVT-B [5]	56.0	8.3	83.2
ViT-Small/16 [5]	48.8	9.9	80.8
HaLViT-M (ours)	**43.0**	**16.8**	**81.3**
T2T-ViTt-24 [47]	64.0	15.0	82.2
PVT-L [40]	61.4	9.8	81.7
TNT-B [13]	66.0	14.1	82.8
SWin-B [29]	88.0	15.4	83.3
Twins-SVT-L [5]	99.2	14.8	83.7
FocalAtt-Small [46]	51.1	9.4	83.5
FocalAtt-Base [46]	89.8	16.4	83.8
PVTv2-B4 [41]	62.6	10.1	83.6
PVTv2-B5 [41]	82.0	11.8	83.8

Table 4. Comparison of ResNet architectures. In ResNet50* our method is applied exclusively to stages 3 and 4.

Model	**Acc@1**	**Acc@5**	**Params (M)**	**GFLOPS**
ResNet18	69.7	89.0	11.7M	1.8
ResNet34	73.3	91.4	21.8M	3.66
ResNet50	76.1	92.8	25.6M	4.09
ResNet50* (Ours)	75.1	92.8	13.4	4.09

Results. Table 3 demonstrates that HaLViT, despite its lower parameter count, achieves performance on par with models such as PVTv2-B1 and DeiT-S/16. Notably, HaLViT^2 outperforms PVTv2-B1, even with a reduced number of parameters. These results suggest that reducing the parameter count by half does not significantly compromise performance, thereby maintaining efficacy with fewer parameters.

Table 4 indicates that the proposed method delivers performance akin to existing benchmarks. Despite sharing

bottleneck layers' parameters in each stage, ResNet50*'s performance nearly matches that of the original ResNet50 and surpasses both ResNet34 and ResNet18. Intriguingly, ResNet50* maintains a parameter count comparable to that of ResNet18.

4.2. Object Detection and Instance Segmentation

Settings. We conduct our object detection experiments using the challenging COCO benchmark. All models are trained on the COCO train2017 dataset, which includes 118k images, and evaluations are carried out on the val2017 dataset, comprising 5k images. To assess the efficiency of HalViT, we integrate it with a conventional detector: Mask R-CNN [17]. We followed the same approach as in [26] for plain backbones. Simple FPN [26] is employed as neck before the Mask R-CNN head. To accommodate images of varying sizes, we adapt the positional embeddings through interpolation. Additionally, padding is applied to ensure that the dimensions of the input images are multiples of 16. At the initiation of training, ImageNet pre-trained weights are utilized to set up the backbone, while the Xavier initialization method is applied to the layers newly introduced. The training process for our models involves a batch size of 8 across 4 A100 GPUs, leveraging AdamW for optimization with an initial learning rate of 0.5×10^{-4}. In line with established practices, models are trained using a 2x training schedule, equivalent to 24 epochs. Training images are resized to maintain a shorter side of 800 pixels and longer side does not exceed 1333 pixels. During testing, the input image's shorter side is consistently set to 800 pixels.

Results. As seen in Table 5, we evaluated the performance of our HaLViT models, HaLViT-T and HaLViT-M against established architectures like ResNet and PVT on the COCO val2017 dataset for object detection and instance segmentation tasks using Mask R-CNN. HaLViT-T, with only 30.8M parameters, demonstrates competitive results, achieving an AP^b of 35.3 and an AP^m of 33.3, underscoring its efficiency. Notably, HaLViT-M, with 63.0M parameters, surpasses other models in its category, achieving superior performance metrics (AP^b: 42.3, AP^m: 39.2), indicating its effectiveness in precise detection and segmentation. The results highlight HaLViT architectures as efficient and effective alternatives for computer vision tasks, marked by their reduced parameter count and competitive performance metrics.

4.3. Transfer Learning

Settings. Despite HaLViT's strong performance on ImageNet, its generalization capability necessitates evaluation via transfer learning on diverse datasets. Accordingly, we conducted fine-tuning experiments with HaLViT on datasets presented in Table 6. The results, compared against ViT and DeiT architectures in Table 7, demonstrate HaLViT's comparability with Transformer models, corroborating our earlier findings on its ImageNet performance.

4.4. Ablation Study

Settings. We conduct ablation studies in ImageNet dataset.The experimental settings on ImageNet are the same as the settings in Sec. 4.1.

Extreme Weight Sharing Across Layers. In our study, we developed a 12-layer HaLViT architecture employing an extreme parameter sharing strategy. This variant, identical to HaLViT-M in structure, utilizes shared parameters across all layers, excluding the query weights, resulting in a significantly reduced model size of only 9M parameters. Results is shown in Table 8. As anticipated, there was a significant decrease in accuracy, yet this outcome underscores that the model can achieve convergence even under conditions of extensive parameter sharing across layers.

Weight Sharing in ResNet Stages. In the ResNet architecture, we implemented a weight sharing approach, utilizing the same weight matrix W and W^T across all 1×1 convolutions within each bottleneck layer at same stages. Results can be seen in Table 9. Implementing weight sharing in the initial stages 1 and 2 only minimally decreases the parameter count while causing a reduction in accuracy by about 0.9 points, indicating that parameter sharing is less beneficial in the early stages.

5. Discussion

We present a novel approach to reduce deep learning model parameters by leveraging the column and row spaces of weight matrices. This approach, focused on Transformers and CNNs, can significantly lower the parameter count without markedly compromising model performance. Our discussion explores the broader applicability and implications of this methodology in various architectures and domains.

Applicability Across Architectures. The foundational principle of HaLViT, utilizing weight matrix spaces for parameter efficiency, presents a versatile strategy extendable beyond ViT and ResNet. Its application could revolutionize parameter efficiency in other architectures as well. This universality underscores the method's potential as a cross-architecture strategy for enhancing parameter efficiency.

Integration with Existing Techniques. Merging HaLViT's approach with pruning and quantization could yield ultra-efficient models ideal for resource-constrained

Table 5. Object detection and instance segmentation performance on COCO val2017 utilizing Mask R-CNN. AP^b and AP^m denote bounding box AP and mask AP, respectively.

Backbone	#Param (M)	AP^b	Ap^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet18 [16]	31.2	34.0	54.0	36.7	31.2	51.0	32.7
PVT-T [40]	32.9	36.7	59.2	39.3	35.1	56.7	37.3
HaLViT-T	**30.8**	**35.3**	**58.6**	**38.5**	**33.3**	**55.2**	**36.2**
ResNet101 [16]	63.2	40.4	61.1	44.2	36.4	57.7	38.8
ResNeXt101-32x4d [45]	62.8	41.9	62.5	45.9	37.5	59.4	40.2
PVT-M [40]	63.9	42.0	64.4	45.6	39.0	61.6	42.1
HaLViT-M	**63.0**	**42.3**	**65.6**	**46.0**	**39.2**	**62.4**	**42.5**

Table 6. Datasets utilized for different tasks.

Dataset	Train size	Test size	#classes
ImageNet [8]	1,281,167	50,000	1000
iNaturalist 2018 [37]	437,513	24,426	8,142
iNaturalist 2019 [37]	265,240	3,003	1,010
Flowers-102 [30]	2,040	6,149	102
Stanford Cars [23]	8,144	8,041	196
CIFAR-100 [1]	50,000	10,000	100
CIFAR-10 [1]	50,000	10,000	10

Table 7. Comparison of Transformers-based models on different transfer learning tasks with ImageNet pre-training and convolutional architectures for reference.

Model	**ImageNet**	**CIFAR-10**	**CIFAR-100**	**Flowers**
ViT-B/32 [9, 36]	73.4	97.8	86.3	85.4
ViT-B/16 [9, 36]	77.9	98.1	87.1	89.5
ViT-L/32 [9, 36]	71.2	97.9	87.1	86.4
ViT-L/16 [9, 36]	76.5	97.9	86.4	89.7
DeiT-B [36]	81.8	99.1	90.8	98.4
HaLViT-T	78.8	98.7	90.3	96.5
HaLViT-M	81.3	99.2	91.0	98.3

Table 8. In HaLViT*, parameter sharing is implemented across all layers, with the exception of the query weights W_Q. Moreover, the described method is also applied in this configuration.

Model	**Acc@1**	**Acc@5**	**Params (M)**	**GFLOPS**
HaLViT-M	81.3	95.6	44.0	16.8
HaLViT-M*	67.6	87.5	9.0	16.8

Table 9. ResNet50[1] utilizes the proposed method across all its stages, whereas ResNet50[2] applies the method to stages 3 and 4.

Model	**Acc@1**	**Acc@5**	**Params (M)**	**GFLOPS**
ResNet50[1]	74.2	91.9	12.8	4.09
ResNet50[2]	75.1	92.8	13.4	4.09

environments. This synergy would enable the deployment of powerful deep learning models on mobile and edge devices, highlighting the method's adaptability to current computational demands.

Enhancements in Federated Learning Proposed method is particularly beneficial in federated learning scenarios, where efficient model communication is paramount. By reducing model size while preserving accuracy, our method could significantly contribute to the scalability and effectiveness of federated learning frameworks.

6. Conclusion

This study presents a novel approach for enhancing parameter efficiency. By exploiting the row and column spaces of the weight matrix $\mathbf{W}$, we successfully reduced the parameters of ViT and ResNet50 by half while maintaining their performance on the ImageNet dataset. The findings suggest that the strategic use of column and row spaces leads to more effective utilization of weight matrices. This method is versatile and can be implemented in any model that employs Transformer layers and bottleneck layers, significantly enhancing its applicability and potency.

7. Acknowledgments

This work was supported by the Scientific and Technological Research Council of Türkiye (TUBITAK) with 1515 Frontier R&D Laboratories Support Program for BTS Advanced AI Hub: BTS Autonomous Networks and Data Innovation Lab. Project 5239903, with grant number 121E378; partly by the Scientific Research Projects Coordination Department (BAP), Istanbul Technical University, under Projects ITU-BAP MGA-2024-45372 and HIZDEP; and in part by the National Center for High Performance Computing (UHEM) with grant numbers 1016682023 and 4016562023.

References

[1] Krizhevsky Alex. Learning multiple layers of features from tiny images. *https://www. cs. toronto. edu/kriz/learning-features-2009-TR. pdf*, 2009. 8

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[3] 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

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 1

[5] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 34:9355–9366, 2021. 5, 6

[6] MMPreTrain Contributors. Openmmlab's pre-training toolbox and benchmark, 2023. 5

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 2

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5, 8

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 6, 8

[10] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019. 6

[11] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 6

[12] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1580–1589, 2020. 3

[13] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34:15908–15919, 2021. 6

[14] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 2

[15] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*, 2021. 1

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 3, 8

[17] 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

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[19] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 3

[20] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 646–661. Springer, 2016. 6

[21] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014. 2

[22] Reyhan Kevser Keser, Aydin Ayanzadeh, Omid Abdollahi Aghdam, Caglar Kilcioglu, Behcet Ugur Toreyin, and Nazim Kemal Ure. Pursuhint: In search of informative hint points based on layer clustering for knowledge distillation. *Expert Systems with Applications*, 213:119040, 2023. 2

[23] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 8

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 1, 3

[25] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989. 3

[26] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *European Conference on Computer Vision*, pages 280–296. Springer, 2022. 7

[27] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural

network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021. 2

[28] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 2022. 2

[29] 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. 1, 6

[30] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE, 2008. 8

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 5

[32] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018. 2

[33] 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. 2

[34] David R So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V Le. Primer: Searching for efficient transformers for language modeling. *arXiv preprint arXiv:2109.08668*, 2021. 2

[35] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 1, 3

[36] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021. 1, 5, 6, 8

[37] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018. 8

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1, 2

[39] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 2

[40] 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. 1, 6, 8

[41] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):415–424, 2022. 6

[42] Ross Wightman et al. Pytorch image models, 2019. 6

[43] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 22–31, 2021. 1, 6

[44] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34: 12077–12090, 2021. 2

[45] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 8

[46] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. *arXiv preprint arXiv:2107.00641*, 2021. 6

[47] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 558–567, 2021. 1, 6

[48] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 6

[49] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 6

[50] Zizhao Zhang, Han Zhang, Long Zhao, Ting Chen, Sercan Ö Arik, and Tomas Pfister. Nested hierarchical transformer: Towards accurate, data-efficient and interpretable visual understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3417–3425, 2022. 1

[51] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 2

[52] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 1