EfficientViT: Memory Efficient Vision Transformer with Cascaded Group Attention

Xinyu Liu[1,*], Houwen Peng[2], Ningxin Zheng[2], Yuqing Yang[2], Han Hu[2], Yixuan Yuan[1]
[1] The Chinese University of Hong Kong, [2] Microsoft Research
1155195604@link.cuhk.edu.hk, {houwen.peng, ningxin.zheng, yuqing.yang, hanhu}@microsoft.com, yxyuan@ee.cuhk.edu.hk

Abstract

Vision transformers have shown great success due to their high model capabilities. However, their remarkable performance is accompanied by heavy computation costs, which makes them unsuitable for real-time applications. In this paper, we propose a family of high-speed vision transformers named EfficientViT. We find that the speed of existing transformer models is commonly bounded by memory inefficient operations, especially the tensor reshaping and element-wise functions in MHSA. Therefore, we design a new building block with a sandwich layout, i.e., using a single memory-bound MHSA between efficient FFN layers, which improves memory efficiency while enhancing channel communication. Moreover, we discover that the attention maps share high similarities across heads, leading to computational redundancy. To address this, we present a cascaded group attention module feeding attention heads with different splits of the full feature, which not only saves computation cost but also improves attention diversity. Comprehensive experiments demonstrate EfficientViT outperforms existing efficient models, striking a good trade-off between speed and accuracy. For instance, our EfficientViT-M5 surpasses MobileNetV3-Large by 1.9% in accuracy, while getting 40.4% and 45.2% higher throughput on Nvidia V100 GPU and Intel Xeon CPU, respectively. Compared to the recent efficient model MobileViT-XXS, EfficientViT-M2 achieves 1.8% superior accuracy, while running 5.8×/3.7× faster on the GPU/CPU, and 7.4× faster when converted to ONNX format. Code and models are available at here.

1. Introduction

Vision Transformers (ViTs) have taken computer vision domain by storm due to their high model capabilities and superior performance [18, 44, 69]. However, the constantly improved accuracy comes at the cost of increasing model sizes and computation overhead. For example, SwinV2 [43] uses 3.0B parameters, while V-MoE [62] taking 14.7B parameters, to achieve state-of-the-art performance on ImageNet [17]. Such large model sizes and the accompanying heavy computational costs make these models unsuitable for applications with real-time requirements [40, 78, 86].

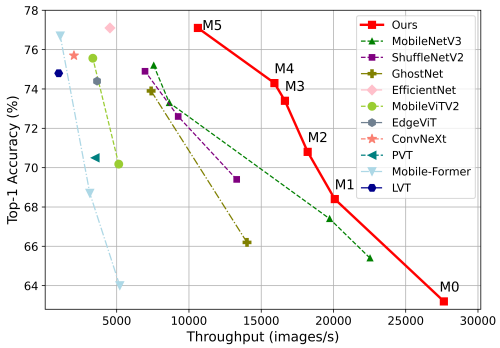

Figure 1. Speed and accuracy comparisons between EfficientViT (Ours) and other efficient CNN and ViT models tested on an Nvidia V100 GPU with ImageNet-1K dataset [17].

There are several recent works designing light and efficient vision transformer models [9, 19, 29, 49, 50, 56, 79, 81]. Unfortunately, most of these methods aim to reduce model parameters or Flops, which are indirect metrics for speed and do not reflect the actual inference throughput of models. For example, MobileViT-XS [50] using 700M Flops runs much slower than DeiT-T [69] with 1,220M Flops on an Nvidia V100 GPU. Although these methods have achieved good performance with fewer Flops or parameters, many of them do not show significant wall-clock speedup against standard isomorphic or hierarchical transformers, *e.g.*, DeiT [69] and Swin [44], and have not gained wide adoption.

To address this issue, in this paper, we explore how to go faster with vision transformers, seeking to find principles for designing efficient transformer architectures. Based on the prevailing vision transformers DeiT [69] and Swin [44], we systematically analyze three main factors that affect model inference speed, including memory access, computation redundancy, and parameter usage. In particular, we find that the speed of transformer models is commonly memory-bound. In other words, memory accessing delay prohibits the full utilization of the computing power in GPU/CPUs [21, 32, 72], leading to a critically negative impact on the runtime speed of transformers [15, 31]. The

*Work done when Xinyu was an intern of Microsoft Research.

most memory-inefficient operations are the frequent tensor reshaping and element-wise functions in multi-head self-attention (MHSA). We observe that through an appropriate adjustment of the ratio between MHSA and FFN (feed-forward network) layers, the memory access time can be reduced significantly without compromising the performance. Moreover, we find that some attention heads tend to learn similar linear projections, resulting in redundancy in attention maps. The analysis shows that explicitly decomposing the computation of each head by feeding them with diverse features can mitigate this issue while improving computation efficiency. In addition, the parameter allocation in different modules is often overlooked by existing lightweight models, as they mainly follow the configurations in standard transformer models [44, 69]. To improve parameter efficiency, we use structured pruning [45] to identify the most important network components, and summarize empirical guidance of parameter reallocation for model acceleration.

Based upon the analysis and findings, we propose a new family of memory efficient transformer models named EfficientViT. Specifically, we design a new block with a sandwich layout to build up the model. The sandwich layout block applies a single memory-bound MHSA layer between FFN layers. It reduces the time cost caused by memory-bound operations in MHSA, and applies more FFN layers to allow communication between different channels, which is more memory efficient. Then, we propose a new cascaded group attention (CGA) module to improve computation efficiency. The core idea is to enhance the diversity of the features fed into the attention heads. In contrast to prior self-attention using the same feature for all heads, CGA feeds each head with different input splits and cascades the output features across heads. This module not only reduces the computation redundancy in multi-head attention, but also elevates model capacity by increasing network depth. Last but not least, we redistribute parameters through expanding the channel width of critical network components such as value projections, while shrinking the ones with lower importance like hidden dimensions in FFNs. This reallocation finally promotes model parameter efficiency.

Experiments demonstrate that our models achieve clear improvements over existing efficient CNN and ViT models in terms of both speed and accuracy, as shown in Fig. 1. For instance, our EfficientViT-M5 gets 77.1% top-1 accuracy on ImageNet with throughput of 10,621 images/s on an Nvidia V100 GPU and 56.8 images/s on an Intel Xeon E5-2690 v4 CPU @ 2.60GHz, outperforming MobileNetV3-Large [26] by 1.9% in accuracy, 40.4% in GPU inference speed, and 45.2% in CPU speed. Moreover, EfficientViT-M2 gets 70.8% accuracy, surpassing MobileViT-XXS [50] by 1.8%, while running 5.8×/3.7× faster on the GPU/CPU, and 7.4× faster when converted to ONNX [3] format. When deployed on the mobile chipset, *i.e.*, Apple A13 Bionic chip

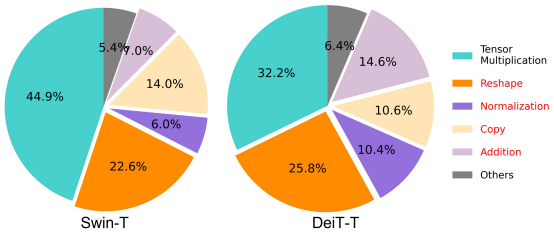

Figure 2. Runtime profiling on two standard vision transformers Swin-T and DeiT-T. Red text denotes memory-bound operations, *i.e.*, the time taken by the operation is mainly determined by memory accesses, while time spent in computation is much smaller.

in iPhone 11, EfficientViT-M2 model runs 2.3× faster than MobileViT-XXS [50] using the CoreML [1].

In summary, the contributions of this work are two-fold:

- We present a systematic analysis on the factors that affect the inference speed of vision transformers, deriving a set of guidelines for efficient model design.
- We design a new family of vision transformer models, which strike a good trade-off between efficiency and accuracy. The models also demonstrate good transfer ability on a variety of downstream tasks.

2. Going Faster with Vision Transformers

In this section, we explore how to improve the efficiency of vision transformers from three perspectives: memory access, computation redundancy, and parameter usage. We seek to identify the underlying speed bottlenecks through empirical studies, and summarize useful design guidelines.

2.1. Memory Efficiency

Memory access overhead is a critical factor affecting model speed [15, 28, 31, 65]. Many operators in transformer [71], such as frequent reshaping, element-wise addition, and normalization are memory inefficient, requiring time-consuming access across different memory units, as shown in Fig. 2. Although there are some methods proposed to address this issue by simplifying the computation of standard softmax self-attention, *e.g.*, sparse attention [34, 57, 61, 75] and low-rank approximation [11, 51, 74], they often come at the cost of accuracy degradation and limited acceleration.

In this work, we turn to save memory access cost by reducing memory-inefficient layers. Recent studies reveal that memory-inefficient operations are mainly located in MHSA rather than FFN layers [31, 33]. However, most existing ViTs [18, 44, 69] use an equivalent number of these two layers, which may not achieve the optimal efficiency. We thereby explore the optimal allocation of MHSA and FFN layers in small models with fast inference. Specifically, we scale down Swin-T [44] and DeiT-T [69] to several small subnetworks with 1.25× and 1.5× higher inference throughput, and compare the performance of subnetworks with different proportions of MHSA layers. As shown in Fig. 3, subnetworks with 20%-40% MHSA layers tend to get better accuracy. Such ratios are much smaller than the

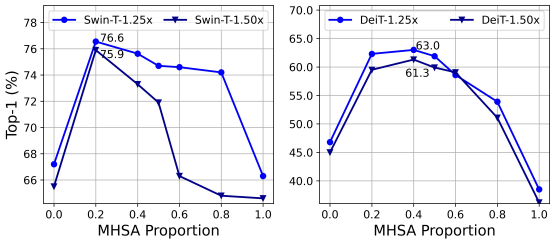

Figure 3. The accuracy of downscaled baseline models with different MHSA layer proportions, where the dots on each line represent subnetworks with similar throughput. **Left:** Swin-T as the baseline. **Right:** DeiT-T as the baseline. The 1.25×/1.5× denote accelerating the baseline models by 1.25/1.5 times, respectively.

typical ViTs that adopt 50% MHSA layers. Furthermore, we measure the time consumption on memory-bound operations to compare memory access efficiency, including reshaping, element-wise addition, copying, and normalization. Memory-bound operations is reduced to 44.26% of the total runtime in Swin-T-1.25× that has 20% MHSA layers. The observation also generalizes to DeiT and smaller models with 1.5× speed-up. It is demonstrated that *reducing MHSA layer utilization ratio appropriately can enhance memory efficiency while improving model performance.*

2.2. Computation Efficiency

MHSA embeds the input sequence into multiple subspaces (heads) and computes attention maps separately, which has been proven effective in improving performance [18, 69, 71]. However, attention maps are computationally expensive, and studies have shown that a number of them are not of vital importance [52, 73]. To save computation cost, we explore how to reduce redundant attention in small ViT models. We train width downscaled Swin-T [44] and DeiT-T [69] models with 1.25× inference speed-up, and measure the maximum cosine similarity of each head and the remaining heads within each block. From Fig. 4, we observe there exists high similarities between attention heads, especially in the last blocks. The phenomenon suggests that many heads learn similar projections of the same full feature and incur computation redundancy. To explicitly encourage the heads to learn different patterns, we apply an intuitive solution by feeding each head with only a split of the full feature, which is similar to the idea of group convolution in [10, 87]. We train the variants of downscaled models with the modified MHSA, and also compute the attention similarities in Fig. 4. It is shown that *using different channel-wise splits of the feature in different heads, instead of using the same full feature for all heads as MHSA, could effectively mitigate attention computation redundancy.*

2.3. Parameter Efficiency

Typical ViTs mainly inherit the design strategies from NLP transformer [71], *e.g.*, using an equivalent width for Q,K,V projections, increasing heads over stages, and setting the expansion ratio to 4 in FFN. For lightweight models, the configurations of these components need to be carefully re-designed [7, 8, 39]. Inspired by [45, 82], we adopt Taylor structured pruning [53] to automatically find the important components in Swin-T and DeiT-T, and explore the underlying principles of parameter allocation. The pruning method removes unimportant channels under a certain resource constraint and keeps the most critical ones to best preserve the accuracy. It uses the multiplication of gradient and weight as channel importance, which approximates the loss fluctuation when removing channels [38].

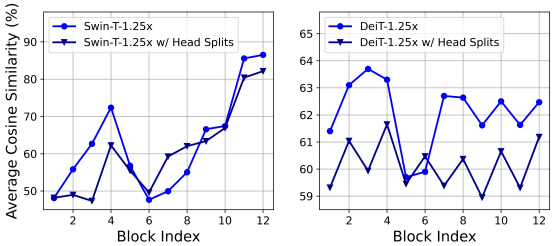

Figure 4. The average maximum cosine similarity of each head in different blocks. **Left:** downscaled Swin-T models. **Right:** downscaled DeiT-T models. Blue lines denote Swin-T-1.25×/DeiT-T-1.25× model, while darkblue lines denote the variants that feed each head with only a split of the full feature.

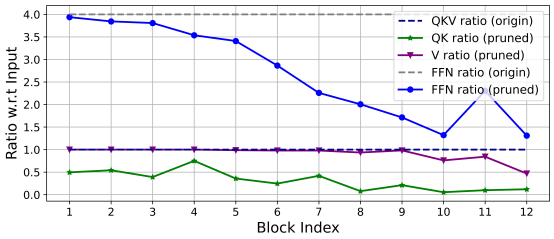

Figure 5. The ratio of the channels to the input embeddings before and after pruning Swin-T. Baseline accuracy: 79.1%; pruned accuracy: 76.5%. Results for DeiT-T are given in the supplementary.

The ratio between the remaining output channels to the input channels is plotted in Fig. 5, and the original ratios in the unpruned model are also given for reference. It is observed that: 1) The first two stages preserve more dimensions, while the last stage keeps much less; 2) The Q,K and FFN dimensions are largely trimmed, whereas the dimension of V is almost preserved and diminishes only at the last few blocks. These phenomena show that *1) the typical channel configuration, that doubles the channel after each stage [44] or use equivalent channels for all blocks [69], may produce substantial redundancy in last few blocks; 2) The redundancy in Q,K is much larger than V when they have the same dimensions. V prefers a relative large channels, being close to the input embedding dimension.*

3. Efficient Vision Transformer

Based upon the above analysis, in this section, we propose a new hierarchical model with fast inference named EfficientViT. The architecture overview is shown in Fig. 6.

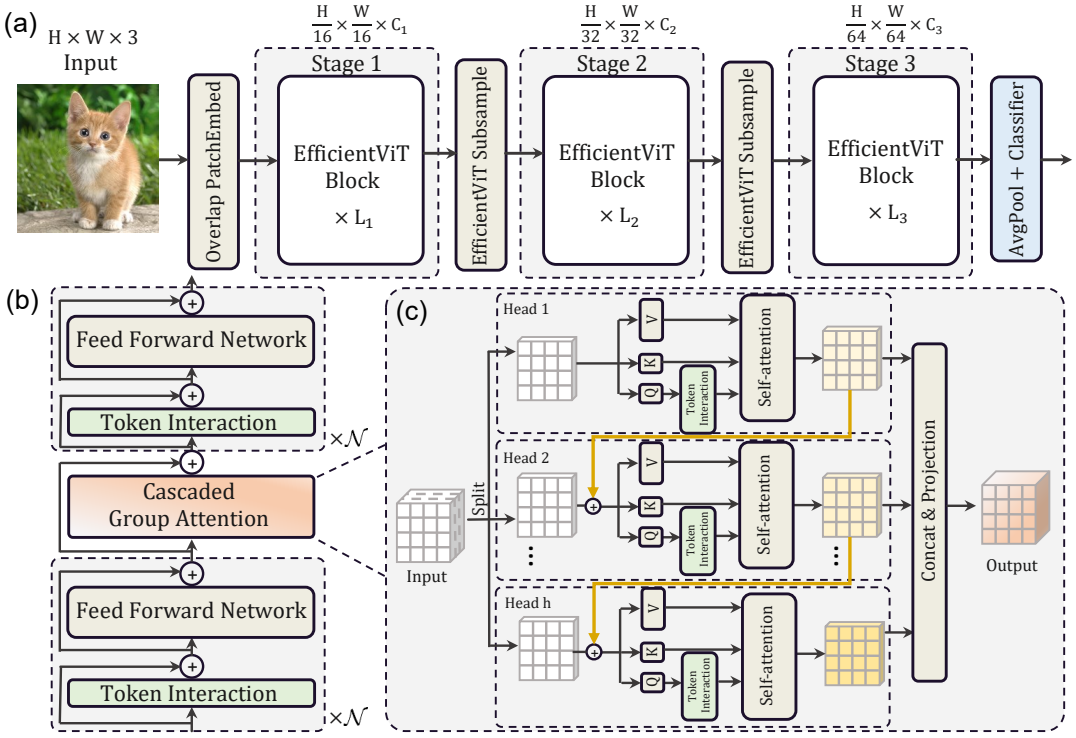

Figure 6. Overview of EfficientViT. (a) Architecture of EfficientViT; (b) Sandwich Layout block; (c) Cascaded Group Attention.

3.1. EfficientViT Building Blocks

We propose a new efficient building block for vision transformer, as shown in Fig. 6 (b). It is composed of a memory-efficient sandwich layout, a cascaded group attention module, and a parameter reallocation strategy, which focus on improving model efficiency in terms of memory, computation, and parameter, respectively.

Sandwich Layout. To build up a memory-efficient block, we propose a sandwich layout that employs less memory-bound self-attention layers and more memory-efficient FFN layers for channel communication. Specifically, it applies a single self-attention layer Φ_i^{A} for spatial mixing, which is sandwiched between FFN layers Φ_i^{F}. The computation can be formulated as:

$$X_{i+1} = \prod^{\mathcal{N}} \Phi_i^{\mathrm{F}}(\Phi_i^{\mathrm{A}}(\prod^{\mathcal{N}} \Phi_i^{\mathrm{F}}(X_i))), \tag{1}$$

where X_i is the full input feature for the i-th block. The block transforms X_i into X_{i+1} with $\mathcal{N}$ FFNs before and after the single self-attention layer. This design reduces the memory time consumption caused by self-attention layers in the model, and applies more FFN layers to allow communication between different feature channels efficiently. We also apply an extra token interaction layer before each FFN using a depthwise convolution (DWConv) [27]. It introduces inductive bias of the local structural information to enhance model capability [14].

Cascaded Group Attention. Attention head redundancy is a severe issue in MHSA, which causes computation inefficiency. Inspired by group convolutions in efficient CNNs [10, 37, 64, 87], we propose a new attention module named cascaded group attention (CGA) for vision transformers. It feeds each head with different splits of the full features, thus explicitly decomposing the attention computation across heads. Formally, this attention can be formulated as:

$$\begin{aligned} \widetilde{X}_{ij} &= Attn(X_{ij}W_{ij}^{\mathrm{Q}}, X_{ij}W_{ij}^{\mathrm{K}}, X_{ij}W_{ij}^{\mathrm{V}}), \\ \widetilde{X}_{i+1} &= Concat[\widetilde{X}_{ij}]_{j=1:h} W_i^{\mathrm{P}}, \end{aligned} \tag{2}$$

where the j-th head computes the self-attention over X_{ij}, which is the j-th split of the input feature X_i, *i.e.*, $X_i = [X_{i1}, X_{i2}, \ldots, X_{ih}]$ and $1 \leq j \leq h$. h is the total number of heads, W_{ij}^{Q}, W_{ij}^{K}, and W_{ij}^{V} are projection layers mapping the input feature split into different subspaces, and W_i^{P} is a linear layer that projects the concatenated output features back to the dimension consistent with the input.

Although using feature splits instead of the full features for each head is more efficient and saves computation overhead, we continue to improve its capacity, by encouraging the Q, K, V layers to learn projections on features with richer information. We compute the attention map of each head in a cascaded manner, as illustrated in Fig. 6 (c), which adds the output of each head to the subsequent head to refine the feature representations progressively:

$$X_{ij}^{'} = X_{ij} + \widetilde{X}_{i(j-1)}, \quad 1 < j \leq h, \tag{3}$$

where $X_{ij}^{'}$ is the addition of the j-th input split X_{ij} and the $(j-1)$-th head output $\widetilde{X}_{i(j-1)}$ calculated by Eq. (2). It replaces X_{ij} to serve as the new input feature for the j-th head when calculating the self-attention. Besides, another token

Table 1. Architecture details of EfficientViT model variants.

Model	$\{C_1, C_2, C_3\}$	$\{L_1, L_2, L_3\}$	$\{H_1, H_2, H_3\}$
EfficientViT-M0	{64, 128, 192}	{1, 2, 3}	{4, 4, 4}
EfficientViT-M1	{128, 144, 192}	{1, 2, 3}	{2, 3, 3}
EfficientViT-M2	{128, 192, 224}	{1, 2, 3}	{4, 3, 2}
EfficientViT-M3	{128, 240, 320}	{1, 2, 3}	{4, 3, 4}
EfficientViT-M4	{128, 256, 384}	{1, 2, 3}	{4, 4, 4}
EfficientViT-M5	{192, 288, 384}	{1, 3, 4}	{3, 3, 4}

interaction layer is applied after the Q projection, which enables the self-attention to jointly capture local and global relations and further enhances the feature representation.

Such a cascaded design enjoys two advantages. First, feeding each head with different feature splits could improve the diversity of attention maps, as validated in Sec. 2.2. Similar to group convolutions [10, 87], the cascaded group attention could save the Flops and parameters by $h\times$, since the input and output channels in the QKV layers are reduced by $h\times$. Second, cascading the attention heads allows for an increase of network depth, thus further elevating the model capacity without introducing any extra parameters. It only incurs minor latency overhead since the attention map computation in each head uses smaller QK channel dimensions.

Parameter Reallocation. To improve parameter efficiency, we reallocate the parameters in the network by expanding the channel width of critical modules while shrinking the unimportant ones. Specifically, based on the Taylor importance analysis in Sec. 2.3, we set small channel dimensions for Q and K projections in each head for all stages. For the V projection, we allow it to have the same dimension as the input embedding. The expansion ratio in FFN is also reduced from 4 to 2 due to its parameter redundancy. With the proposed reallocation strategy, the important modules have larger number of channels to learn representations in a high dimensional space, which prevent the loss of feature information. Meanwhile, the redundant parameters in unimportant modules are removed to speed up inference and enhance the model efficiency.

3.2. EfficientViT Network Architectures

The overall architecture of our EfficientViT is presented in Fig. 6 (a). Concretely, we introduce overlapping patch embedding [20, 80] to embed 16×16 patches into tokens with C_1 dimension, which enhances the model capacity in low-level visual representation learning. The architecture contains three stages. Each stage stacks the proposed EfficientViT building blocks and the number of tokens is reduced by 4× at each subsampling layer (2× subsampling of the resolution). To achieve efficient subsampling, we propose an EfficientViT subsample block which also has the sandwich layout, except that the self-attention layer is replaced by an inverted residual block to reduce the information loss during subsampling [26, 63]. It is worth noting that we adopt BatchNorm (BN) [30] instead of LayerNorm (LN) [2] throughout the model, as BN can be folded into the preceding convolution or linear layers, which is a runtime advantage over LN. We also use ReLU [54] as the activation function, as the commonly used GELU [25] or HardSwish [26] are much slower, and sometimes not well-supported by certain inference deployment platforms [1, 3].

We build our model family with six different width and depth scales, and set different number of heads for each stage. We use fewer blocks in early stages than late stages similar to MobileNetV3 [26] and LeViT [20], since that the processing on early stages with larger resolutions is more time consuming. We increase the width over stages with a small factor (≤ 2) to alleviate redundancy in later stages, as analyzed in Sec. 2.3. The architecture details of our model family are presented in Tab. 1. C_i, L_i, and H_i refer to the width, depth, and number of heads in the i-th stage.

4. Experiments

4.1. Implementation Details

We conduct image classification experiments on ImageNet-1K [17]. The models are built with PyTorch 1.11.0 [59] and Timm 0.5.4 [77], and trained from scratch for 300 epochs on 8 Nvidia V100 GPUs using AdamW [46] optimizer and cosine learning rate scheduler. We set the total batchsize as 2,048. The input images are resized and randomly cropped into 224×224. The initial learning rate is 1×10^{-3} with weight decay of 2.5×10^{-2}. We use the same data augmentation as [69], including Mixup [85], auto-augmentation [13], and random erasing [88]. In addition, we provide throughput evaluation on different hardware. For GPU, we measure the throughput on an Nvidia V100, with the maximum power-of-two batchsize that fits in memory following [20, 69]. For CPU and ONNX, we measure the runtime on an Intel Xeon E5-2690 v4 @ 2.60 GHz processor, with batchsize 16 and run the model in a single thread following [20]. We also test the transferability of EfficientViT on downstream tasks. For the experiments on downstream image classification, we finetune the models for 300 epochs following [86], using AdamW [46] with batchsize 256, learning rate 1×10^{-3} and weight-decay 1×10^{-8}. We use RetinaNet [41] for object detection on COCO [42], and train the models for 12 epochs (1× schedule) with the same settings as [44] on mmdetection [6]. For instance segmentation, please refer to the supplementary.

4.2. Results on ImageNet

We compare EfficientViT with prevailing efficient CNN and ViT models on ImageNet [17], and report the results in Tab. 2 and Fig. 1. The results show that, in most cases, our EfficientViT achieves the best accuracy and speed trade-off across different evaluation settings.

Comparisons with efficient CNNs. We first compare EfficientViT with vanilla CNN models, such as MobileNets

Table 2. EfficientViT image classification performance on ImageNet-1K [17] with comparisons to state-of-the-art efficient CNN and ViT models trained without extra data. Throughput is tested on Nvidia V100 for GPU and Intel Xeon E5-2690 v4 @ 2.60 GHz processor for CPU and ONNX, where larger throughput means faster inference speed. ↑: finetune with higher resolution.

Model	Top-1 (%)	Top-5 (%)	Throughput (images/s)			Flops (M)	Params (M)	Input	Epochs
			GPU	CPU	ONNX				
EfficientViT-M0	**63.2**	85.4	27644	228.4	340.1	79	2.3	224	300
MobileNetV3-Small [26]	67.4	-	19738	156.5	231.7	57	2.5	224	600
EfficientViT-M1	**68.4**	88.7	20093	126.9	215.9	167	3.0	224	300
Mobile-Former-52M [9]	68.7	-	3141	32.8	21.5	52	3.5	224	450
MobileViT-XXS [50]	69.0	-	4456	29.4	41.7	410	1.3	256	300
ShuffleNetV2 1.0× [48]	69.4	88.9	13301	106.7	177.0	146	2.3	224	300
MobileViTV2-0.5 [51]	70.2	-	5142	34.4	44.9	466	1.4	256	300
EfficientViT-M2	**70.8**	90.2	18218	121.2	158.7	201	4.2	224	300
MobileOne-S0 [70]	71.4	-	11320	67.4	128.6	274	2.1	224	300
MobileNetV2 1.0× [63]	72.0	91.0	6534	32.5	80.4	300	3.4	224	300
EfficientViT-M3	**73.4**	91.4	16644	96.4	120.8	263	6.9	224	300
GhostNet 1.0× [23]	73.9	91.4	7382	57.3	77.0	141	5.2	224	300
NASNet-A-Mobile [89]	74.1	-	2623	19.8	25.5	564	5.3	224	300
EfficientViT-M4	**74.3**	91.8	15914	88.5	108.6	299	8.8	224	300
EdgeViT-XXS [56]	74.4	-	3638	28.2	29.6	556	4.1	224	300
MobileViT-XS [50]	74.7	-	3344	11.1	20.5	986	2.3	256	300
ShuffleNetV2 2.0× [48]	74.9	92.4	6962	37.9	52.3	591	7.4	224	300
MobileNetV3-Large [26]	75.2	-	7560	39.1	70.5	217	5.4	224	600
MobileViTV2-0.75 [51]	75.6	-	3350	16.0	22.7	1030	2.9	256	300
MobileOne-S1 [70]	75.9	-	6663	30.7	51.1	825	4.8	224	300
GLiT-Tiny [5]	76.4	-	3516	17.5	15.7	1333	7.3	224	300
EfficientNet-B0 [67]	77.1	93.3	4532	30.2	29.5	390	5.3	224	350
EfficientViT-M5	**77.1**	93.4	10621	56.8	62.5	522	12.4	224	300
EfficientViT-M4↑384	**79.8**	95.0	3986	15.8	22.6	1486	12.4	384	330
EfficientViT-M5↑512	**80.8**	95.5	2313	8.3	10.5	2670	12.4	512	360

[26, 63] and EfficientNet [67]. Specifically, compared to MobileNetV2 1.0× [63], EfficientViT-M3 obtains 1.4% better top-1 accuracy, while running at 2.5× and 3.0× faster speed on V100 GPU and Intel CPU, respectively. Compared to the state-of-the-art MobileNetV3-Large [26], EfficientViT-M5 achieves 1.9% higher accuracy yet runs much faster, *e.g.*, 40.5% faster on the V100 GPU and 45.2% faster on the Intel CPU but is 11.5% slower as ONNX models. This may because reshaping is slower in ONNX implementation, which is inevitable in computing self-attention. Moreover, EfficientViT-M5 achieves comparable accuracy with the searched model EfficientNet-B0 [67], while runs 2.3×/1.9× faster on the V100 GPU/Intel CPU, and 2.1× faster as ONNX models. Although our model uses more parameters, it reduces memory-inefficient operations that affect the inference speed and achieves higher throughput.

Comparisons with efficient ViTs. We also compare our models with recent efficient vision transformers [5, 9, 50, 51, 56] in Tab. 2. In particular, when getting similar performance on ImageNet-1K [17], our EfficientViT-M4 runs 4.4× and 3.0× faster than the recent EdgeViT-XXS [56] on the tested CPU and GPU devices. Even converted to ONNX runtime format, our model still gets 3.7× higher speed. Compared to the state-of-the-art MobileViTV2-0.5 [51], our EfficientViT-M2 achieves slightly better performance with higher throughput, *e.g.*, 3.4× and 3.5× higher throughput tested on the GPU and CPU devices, respectively. Furthermore, we compare with tiny variants of state-of-the-art large ViTs in Tab. 3. PoolFormer-12S [83] has comparable accuracy with EfficientViT-M5 yet runs 3.0× slower on the V100 GPU. Compared to Swin-T [44], EfficientViT-M5 is 4.1% inferior in accuracy yet is 12.3× faster on the Intel CPU, demonstrating the efficiency of the proposed design. In addition, we present the speed evaluation and comparison on mobile chipsets in the supplementary material.

Finetune with higher resolutions. Recent works on ViTs have demonstrated that finetuning with higher resolutions can further improve the capacity of the models. We also finetune our largest model EfficientViT-M5 to higher resolutions. EfficientViT-M5↑384 reaches 79.8% top-1 accuracy with throughput of 3,986 images/s on the V100 GPU, and EfficientViT-M5↑512 further improves the top-1 accuracy to 80.8%, demonstrating the efficiency on processing images with larger resolutions and the good model capacity.

Table 3. Comparison with the tiny variants of state-of-the-art large-scale ViTs on ImageNet-1K [17].

Model	Top-1 (%)	Throughput (imgs/s) GPU	CPU	ONNX	Flops (G)	Params (M)
PVTV2-B0 [76]	70.5	3507	12.7	18.5	0.6	1.4
T2T-ViT-7 [84]	71.7	1156	22.5	16.1	1.1	4.3
DeiT-T [69]	72.2	4631	26.0	25.1	1.3	5.9
PoolFormer-12S [83]	77.2	3534	10.4	14.6	1.9	12.0
EffFormer-L1 [40]	79.2	4465	12.9	21.2	1.3	12.3
Swin-T [44]	**81.2**	1393	4.6	6.4	4.5	29.0
EfficientViT-M5	77.1	**10621**	**56.8**	**62.5**	0.5	12.4

Table 4. Results of EfficientViT and other efficient models on downstream image classification datasets.

Model	Throughput	ImageNet	CIFAR10	CIFAR100	Cars	Flowers	Pets
MobileNetV1 [27]	8543	70.6	96.1	82.3	**91.4**	96.7	89.9
MobileNetV2 [63]	6534	72.9	95.7	80.8	91.0	96.6	90.5
MobileNetV3 [26]	7560	75.2	97.6	85.5	91.2	97.0	90.1
NASNet-A-M [89]	2623	74.1	96.8	83.9	88.5	96.8	89.4
ViT-S/16 [18]	2135	**81.4**	97.6	85.7	-	86.4	90.4
EfficientViT-M5	**10621**	77.1	**98.0**	**86.4**	89.7	**97.1**	**92.0**

4.3. Transfer Learning Results

To further evaluate the transfer ability, we apply EfficientViT on various downstream tasks.

Downstream Image Classification. We transfer EfficientViT to downstream image classification datasets to test its generalization ability: 1) CIFAR-10 and CIFAR-100 [36]; 2) fine-grained classification: Flowers [55], Stanford Cars [35], and Oxford-IIIT Pets [58]. We report the results in Tab. 4. Compared to existing efficient models [18, 26, 27, 63, 89], our EfficientViT-M5 achieves comparable or slightly better accuracy across all datasets with much higher throughput. An exception lies in Cars, where our model is slightly inferior in accuracy. This may because the subtle differences between classes lie more in local details thus is more feasible to be captured with convolution.

Object Detection. We compare EfficientViT-M4 with efficient models [12, 22, 26, 63, 66, 68] on the COCO [42] object detection task, and present the results in Tab. 5. Specifically, EfficientViT-M4 surpasses MobileNetV2 [63] by 4.4% AP with comparable Flops. Compared to the searched method SPOS [22], our EfficientViT-M4 uses 18.1% fewer Flops while achieving 2.0% higher AP, demonstrating its capacity and generalization ability in different vision tasks.

4.4. Ablation Study

In this section, we ablate important design elements in the proposed EfficientViT on ImageNet-1K [17]. All models are trained for 100 epochs to magnify the differences and reduce training time [20]. Tab. 6 reports the results.

Impact of the sandwich layout block. We first present an ablation study to verify the efficiency of the proposed sandwich layout design, by replacing the sandwich layout block with the original Swin block [44]. The depth is adjusted to $\{2, 2, 3\}$ to guarantee similar throughput with EfficientViT-M4 for a fair comparison. The top-1 accuracy degrades by 3.0% at a similar speed, verifying that applying more FFNs instead of memory-bound MHSA is more effective for small models. Furthermore, to analyze the impact of the number of FFNs $\mathcal{N}$ before and after self-attention , we change the number from 1 to 2 and 3. The number of blocks is reduced accordingly to maintain similar throughput. As presented in Tab. 6 (#3 and #4), further increasing the number of FFNs is not effective due to the lack of long-range spatial relation and $\mathcal{N}$=1 achieves the best efficiency.

Table 5. EfficientViT object detection performance on COCO `val2017` [42] with comparisons to other efficient models.

Model	RetinaNet 1× AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	Flops (M)	Params (M)
MobileNetV2 [63]	28.3	46.7	29.3	14.8	30.7	38.1	300	3.4
MobileNetV3 [26]	29.9	49.3	30.8	14.9	33.3	41.1	217	5.4
SPOS [22]	30.7	49.8	32.2	15.4	33.9	41.6	365	4.3
MNASNet-A2 [66]	30.5	50.2	32.0	16.6	34.1	41.1	340	4.8
FairNAS-C [12]	31.2	50.8	32.7	16.3	34.4	42.3	325	5.6
MixNet-M [68]	31.3	51.7	32.4	17.0	35.0	41.9	360	5.0
EfficientViT-M4	**32.7**	**52.2**	**34.1**	**17.6**	**35.3**	**46.0**	299	8.8

Table 6. Ablation for EfficientViT-M4 on ImageNet-1K [17] dataset. Top-1 accuracy, GPU and ONNX throughput are reported.

#	Ablation	Top-1 (%)	Throughput (imgs/s) GPU	ONNX
1	EfficientViT-M4	71.3	15914	108.6
2	Sandwich → Swin [44]	68.3	15804	114.5
3	$\mathcal{N} = 1 \rightarrow 2$	70.2	14977	112.3
4	$\mathcal{N} = 1 \rightarrow 3$	65.7	15856	139.7
5	CGA → MHSA [18]	70.2	16243	102.2
6	Cascade → None	69.8	16411	111.0
7	QKV allocation → None	69.9	15132	103.1
8	FFN ratio 2 → 4	69.8	15310	112.4
9	DWConv → None	69.9	16325	110.4
10	BN → LN [2]	70.4	15463	103.6
11	ReLU → HSwish [26]	72.2	15887	87.5

Impact of the cascaded group attention. We have proposed CGA to improve the computation efficiency of MHSA. As shown in Tab. 6 (#5 and #6), replacing CGA with MHSA decreases the accuracy by 1.1% and ONNX speed by 5.9%, suggesting that addressing head redundancy improves the model efficiency. For the model without the cascade operation, its performance is comparable with MHSA but worse than CGA, demonstrating the efficacy of enhancing the feature representations of each head.

Impact of the parameter reallocation. Our EfficientViT-M4 yields 1.4%/1.5% higher top-1 accuracy, 4.9%/3.8% higher GPU throughput than the models without QKV channel dimension reallocation or FFN ratio reduction, respectively, indicating the effectiveness of parameter reallocation (#1 *vs.* #7, #8). Moreover, we study the choices of QK dimension in each head and the ratio of V dimension to the input embedding in Fig. 7. It is shown that the performance is improved gradually as QK dimension increases

Table 7. Performance comparison on ImageNet-1K [17] and ImageNet-ReaL [4]. Results with † are trained with 1,000 epochs and knowledge distillation following LeViT [20].

Model	ImageNet (%)			Throughput (imgs/s)			Flops	Params
	Top-1	Top-1†	ReaL†	GPU	CPU	ONNX	(M)	(M)
LeViT-128S [20]	73.6	76.6	82.6	14457	82.3	80.9	305	**7.8**
EfficientViT-M4	**74.3**	**77.1**	**83.6**	**15914**	**88.5**	**108.6**	**299**	8.8

from 4 to 16, while further increasing it gives inferior performance. Besides, the performance improves from 70.3% to 71.3% when increasing the ratio between V dimension and input embedding from 0.4 to 1.0. When further enlarging the ratio to 1.2, it only gets 0.1% improvements. Therefore, setting the channels of V close to the input embedding achieves the best parameter efficiency, which meets our analysis in Sec. 2.3 and design strategy.

Impact of other components. We ablate the impact of using DWConv for token interaction, the normalization layer, and the activation function, as presented in Tab. 6 (#9, #10, and #11). With DWConv, the accuracy improves by 1.4% with a minor latency overhead, demonstrating the effectiveness of introducing local structural information. Replacing BN with LN decreases accuracy by 0.9% and GPU speed by 2.9%. Using HardSwish instead of ReLU improves accuracy by 0.9% but leads to a large drop of 20.0% ONNX speed. The activation functions are element-wise operations that occupy a considerable amount of processing time on GPU/CPU [15, 48, 72], thus utilizing ReLU instead of more complicated activation functions is of better efficiency.

Results of 1,000 training epochs and distillation. Tab. 7 shows the results with 1,000 training epochs and knowledge distillation using RegNetY-16GF [60] as the teacher model following [20] on ImageNet-1K [17] and ImageNet-ReaL [4]. Compared to LeViT-128S [20], EfficientViT-M4 surpasses it by 0.5% on ImageNet-1K and 1.0% on ImageNet-ReaL, respectively. For the inference speed, our model has 34.2% higher throughput on ONNX and also shows superiority on other settings. The results demonstrate that the strong capability and generalization ability of EfficientViT can be further explored with longer training schedules.

5. Related Work

Efficient CNNs. With the demand of deploying CNNs on resource-constrained scenarios, efficient CNNs have been intensively studied in literature [23, 24, 26, 48, 63, 67, 87]. Xception [10] proposes an architecture built with depthwise separable convolutions. MobileNetV2 [63] builds an inverted residual structure which expands the input to a higher dimension. MobileNetV3 [26] and EfficientNet [67] resort to neural architecture search techniques to design compact models. To boost the actual speed on hardware, ShuffleNetV2 [48] introduces channel split and shuffle operations to improve the information communication among channel groups. However, the spatial locality of convolutional kernels hampers CNN models from capturing long-range dependencies, thus limiting their model capacity.

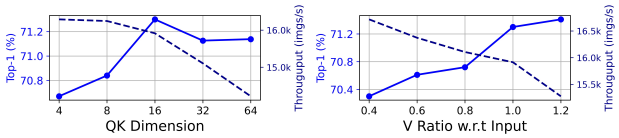

Figure 7. Ablation on the QK dimension of each head and the ratio of V dimension to the input embedding.

Efficient ViTs. ViT and its variants [18, 44, 69, 76] have achieved success on various vision tasks. Despite the superior performance, most of them are inferior to typical CNNs in inference speed. Some efficient transformers have been proposed recently and they fall into two camps: 1) efficient self-attention; and 2) efficient architecture design. Efficient self-attention methods reduce the cost of softmax attention via sparse attention [34, 57, 61, 75] or low-rank approximation [11, 51, 74]. However, they suffer from performance degradation with negligible or moderate inference acceleration over softmax attention [71]. Another line of work combines ViTs with lightweight CNNs to build efficient architectures [9, 47, 49, 50, 81]. LVT [81] proposes enhanced attention mechanisms with dilated convolution to improve the model performance and efficiency. Mobile-Former [9] designs a parallel CNN-transformer block to encode both local features and global interaction. However, most of them target at minimizing Flops and parameters [16], which could have low correlations with actual inference latency [70] and still inferior to efficient CNNs in speed. Different from them, we explore models with fast inference by directly optimizing their throughput on different hardware and deployment settings, and design a family of hierarchical models with a good trade-off between speed and accuracy.

6. Conclusion

In this paper, we have presented a systematic analysis on the factors that affect the inference speed of vision transformers, and proposed a new family of fast vision transformers with memory-efficient operations and cascaded group attention, named EfficientViT. Extensive experiments have demonstrated the efficacy and high speed of EfficientViT, and also show its superiority on various downstream benchmarks.

Limitations. One limitation of EfficientViT is that, despite its high inference speed, the model size is slightly larger compared to state-of-the-art efficient CNN [26] due to the extra FFNs in the introduced sandwich layout. Besides, our models are designed manually based on the derived guidelines on building efficient vision transformers. In future work, we are interested in reducing the model size and incorporating automatic search techniques to further enhance the model capacity and efficiency.

Acknowledgement. Prof. Yuan was partially supported by Hong Kong Research Grants Council (RGC) General Research Fund 11211221, and Innovation and Technology Commission-Innovation and Technology Fund ITS/100/20.

References

[1] Apple. Coremltools: Use coremltools to convert machine learning models from third-party libraries to the core ml format, 2021. 2, 5

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv*, 2016. 5, 7

[3] Junjie Bai, Fang Lu, Ke Zhang, et al. Onnx: Open neural network exchange. https://github.com/onnx/onnx, 2019. 2, 5

[4] Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020. 8

[5] Boyu Chen, Peixia Li, Chuming Li, Baopu Li, Lei Bai, Chen Lin, Ming Sun, Junjie Yan, and Wanli Ouyang. Glit: Neural architecture search for global and local image transformer. In *ICCV*, 2021. 6

[6] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 5

[7] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *ICCV*, 2021. 3

[8] Wuyang Chen, Wei Huang, Xianzhi Du, Xiaodan Song, Zhangyang Wang, and Denny Zhou. Auto-scaling vision transformers without training. In *ICLR*, 2021. 3

[9] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-former: Bridging mobilenet and transformer. In *CVPR*, 2022. 1, 6, 8

[10] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017. 3, 4, 5, 8

[11] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *ICLR*, 2021. 2, 8

[12] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *ICCV*, pages 12239–12248, 2021. 7

[13] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, pages 113–123, 2019. 5

[14] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *NeurIPS*, 34:3965–3977, 2021. 4

[15] Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *arXiv preprint arXiv:2205.14135*, 2022. 1, 2, 8

[16] Mostafa Dehghani, Anurag Arnab, Lucas Beyer, Ashish Vaswani, and Yi Tay. The efficiency misnomer. In *ICLR*, 2022. 8

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 5, 6, 7, 8

[18] 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. *ICLR*, 2021. 1, 2, 3, 7, 8

[19] Chengyue Gong, Dilin Wang, Meng Li, Xinlei Chen, Zhicheng Yan, Yuandong Tian, qiang liu, and Vikas Chandra. NASVit: Neural architecture search for efficient vision transformers with gradient conflict aware supernet training. In *ICLR*, 2022. 1

[20] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *ICCV*, 2021. 5, 7, 8

[21] Jiaqi Gu, Hanqing Zhu, Chenghao Feng, Mingjie Liu, Zixuan Jiang, Ray T Chen, and David Z Pan. Towards memory-efficient neural networks via multi-level in situ generation. In *ICCV*, pages 5229–5238, 2021. 1

[22] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, pages 544–560. Springer, 2020. 7

[23] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *CVPR*, pages 1580–1589, 2020. 6, 8

[24] Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chunjing Xu, Enhua Wu, and Qi Tian. Ghostnets on heterogeneous devices via cheap operations. *Int. J. Comput. Vis.*, 130(4):1050–1069, 2022. 8

[25] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv*, 2016. 5

[26] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *ICCV*, 2019. 2, 5, 6, 7, 8

[27] 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. 4, 7

[28] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. Transformer quality in linear time. In *ICML*, pages 9099–9117. PMLR, 2022. 2

[29] Tao Huang, Lang Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. Lightvit: Towards light-weight convolution-free vision transformers. *arXiv preprint arXiv:2207.05557*, 2022. 1

[30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[31] Andrei Ivanov, Nikoli Dryden, Tal Ben-Nun, Shigang Li, and Torsten Hoefler. Data movement is all you need: A case study on optimizing transformers. *MLSys*, 3:711–732, 2021. 1, 2

[32] Congfeng Jiang, Yitao Qiu, Weisong Shi, Zhefeng Ge, Jiwei Wang, Shenglei Chen, Christophe Cerin, Zujie Ren, Guoyao

Xu, and Jiangbin Lin. Characterizing co-located workloads in alibaba cloud datacenters. *IEEE Trans. on Cloud Comput.*, 2020. 1

[33] Sheng-Chun Kao, Suvinay Subramanian, Gaurav Agrawal, and Tushar Krishna. An optimized dataflow for mitigating attention performance bottlenecks. *arXiv preprint arXiv:2107.06419*, 2021. 2

[34] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020. 2, 8

[35] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCVW*, pages 554–561, 2013. 7

[36] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 7

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, 2017. 4

[38] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *NeurIPS*, 2, 1989. 3

[39] Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Ds-net++: Dynamic weight slicing for efficient inference in cnns and vision transformers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022. 3

[40] Yanyu Li, Geng Yuan, Yang Wen, Eric Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. In *NeurIPS*, 2022. 1, 7

[41] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017. 5

[42] 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 *ECCV*, 2014. 5, 7

[43] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *CVPR*, 2022. 1

[44] 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 *ICCV*, 2021. 1, 2, 3, 5, 6, 7, 8

[45] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *ICLR*, 2018. 2, 3

[46] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2018. 5

[47] Gen Luo, Yiyi Zhou, Xiaoshuai Sun, Yan Wang, Liujuan Cao, Yongjian Wu, Feiyue Huang, and Rongrong Ji. Towards lightweight transformer via group-wise transformation for vision-and-language tasks. *IEEE Trans. Image Process.*, 31:3386–3398, 2022. 8

[48] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, pages 116–131, 2018. 6, 8

[49] Muhammad Maaz, Abdelrahman Shaker, Hisham Cholakkal, Salman Khan, Syed Waqas Zamir, Rao Muhammad Anwer, and Fahad Shahbaz Khan. Edgenext: efficiently amalgamated cnn-transformer architecture for mobile vision applications. In *ECCVW*, 2022. 1, 8

[50] Sachin Mehta and Mohammad Rastegari. Mobilevit: Lightweight, general-purpose, and mobile-friendly vision transformer. In *ICLR*, 2021. 1, 2, 6, 8

[51] Sachin Mehta and Mohammad Rastegari. Separable self-attention for mobile vision transformers. *arXiv preprint arXiv:2206.02680*, 2022. 2, 6, 8

[52] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *NeurIPS*, 32, 2019. 3

[53] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *CVPR*, pages 11264–11272, 2019. 3

[54] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 5

[55] M-E Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *CVPR*, 2006. 7

[56] Junting Pan, Adrian Bulat, Fuwen Tan, Xiatian Zhu, Lukasz Dudziak, Hongsheng Li, Georgios Tzimiropoulos, and Brais Martinez. Edgevits: Competing light-weight cnns on mobile devices with vision transformers. In *ECCV*, 2022. 1, 6

[57] Zizheng Pan, Jianfei Cai, and Bohan Zhuang. Fast vision transformers with hilo attention. *NeurIPS*, 2022. 2, 8

[58] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, pages 3498–3505, 2012. 7

[59] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 2019. 5

[60] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, pages 10428–10436, 2020. 8

[61] Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. Combiner: Full attention transformer with sparse computation cost. *NeurIPS*, 34:22470–22482, 2021. 2, 8

[62] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *NeurIPS*, 34:8583–8595, 2021. 1

[63] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018. 5, 6, 7, 8

[64] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016. 4

[65] Hamid Tabani, Ajay Balasubramaniam, Shabbir Marzban, Elahe Arani, and Bahram Zonooz. Improving the efficiency of transformers for resource-constrained devices. In *DSD*, pages 449–456, 2021. 2

[66] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, pages 2820–2828, 2019. 7

[67] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 6, 8

[68] Mingxing Tan and Quoc V Le. Mixconv: Mixed depthwise convolutional kernels. In *BMVC*, 2019. 7

[69] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*. PMLR, 2021. 1, 2, 3, 5, 7, 8

[70] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. An improved one millisecond mobile backbone. *arXiv preprint arXiv:2206.04040*, 2022. 6, 8

[71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2, 3, 8

[72] Anand Venkat, Tharindu Rusira, Raj Barik, Mary Hall, and Leonard Truong. Swirl: High-performance many-core cpu code generation for deep neural networks. *Int. J. High Perform. Comput. Appl.*, 33(6):1275–1289, 2019. 1, 8

[73] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL*, pages 5797–5808, 2019. 3

[74] 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, 8

[75] 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 *ICCV*, 2021. 2, 8

[76] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvtv2: Improved baselines with pyramid vision transformer. *Comput. Vis. Media*, 2022. 7, 8

[77] Ross Wightman. Pytorch image models, 2019. 5

[78] Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In *ECCV*, 2022. 1

[79] Sitong Wu, Tianyi Wu, Haoru Tan, and Guodong Guo. Pale transformer: A general vision transformer backbone with pale-shaped attention. In *AAAI*, 2022. 1

[80] Tete Xiao, Piotr Dollar, Mannat Singh, Eric Mintun, Trevor Darrell, and Ross Girshick. Early convolutions help transformers see better. *NeurIPS*, 2021. 5

[81] Chenglin Yang, Yilin Wang, Jianming Zhang, He Zhang, Zijun Wei, Zhe Lin, and Alan Yuille. Lite vision transformer with enhanced self-attention. In *CVPR*, pages 11998–12008, 2022. 1, 8

[82] Huanrui Yang, Hongxu Yin, Pavlo Molchanov, Hai Li, and Jan Kautz. Nvit: Vision transformer compression and parameter redistribution. *arXiv preprint arXiv:2110.04869*, 2021. 3

[83] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *CVPR*, pages 10819–10829, 2022. 6, 7

[84] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *ICCV*, 2021. 7

[85] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 5

[86] Jinnian Zhang, Houwen Peng, Kan Wu, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Minivit: Compressing vision transformers with weight multiplexing. In *CVPR*, 2022. 1, 5

[87] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018. 3, 4, 5, 8

[88] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, 2020. 5

[89] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pages 8697–8710, 2018. 6, 7