

SHViT: Single-Head Vision Transformer with Memory Efficient Macro Design

Seokju Yun Youngmin Ro*

Machine Intelligence Laboratory, University of Seoul, Korea

Code: <https://github.com/ysj9909/SHViT>

Abstract

Recently, efficient Vision Transformers have shown great performance with low latency on resource-constrained devices. Conventionally, they use 4×4 patch embeddings and a 4-stage structure at the macro level, while utilizing sophisticated attention with multi-head configuration at the micro level. This paper aims to address computational redundancy at all design levels in a memory-efficient manner. We discover that using larger-stride patchify stem not only reduces memory access costs but also achieves competitive performance by leveraging token representations with reduced spatial redundancy from the early stages. Furthermore, our preliminary analyses suggest that attention layers in the early stages can be substituted with convolutions, and several attention heads in the latter stages are computationally redundant. To handle this, we introduce a single-head attention module that inherently prevents head redundancy and simultaneously boosts accuracy by parallelly combining global and local information. Building upon our solutions, we introduce SHViT, a Single-Head Vision Transformer that obtains the state-of-the-art speed-accuracy tradeoff. For example, on ImageNet-1k, our SHViT-S4 is $3.3\times$, $8.1\times$, and $2.4\times$ faster than MobileViTv2 $\times 1.0$ on GPU, CPU, and iPhone12 mobile device, respectively, while being 1.3% more accurate. For object detection and instance segmentation on MS COCO using Mask-RCNN head, our model achieves performance comparable to FastViT-S412 while exhibiting $3.8\times$ and $2.0\times$ lower backbone latency on GPU and mobile device, respectively.

1. Introduction

Vision Transformers (ViT) have demonstrated impressive performance across various computer vision tasks due to their high model capabilities [1–3]. Compared to Convolutional Neural Networks (CNN) [4, 5], ViTs excel in modeling long-range dependencies and scale effectively with large amounts of training data and model parameters [6]. Despite these advantages, the lack of inductive bias in

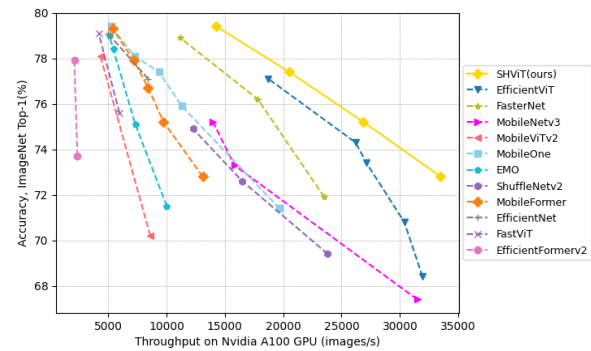


Figure 1. Comparison of throughput and accuracy between our SHViT and other recent methods.

vanilla ViTs necessitates more training data, and the global attention module incurs quadratic computational complexity with respect to the image size. To address these challenges, previous research has either combined ViTs with CNNs or introduced cost-efficient attention variants.

Recently, studies addressing problems with real-time constraints have also proposed efficient models following similar strategies. And their strategies can be categorized into two groups: 1) efficient architecture - *macro design*; and 2) efficient Multi-Head Self-Attention (MHSA) - *micro design*. Studies exploring architectural design [7–12] utilize convolution to handle high-resolution / low-level features and employ attention for low-resolution / high-level features, demonstrating superior performance without complex operations. However, most of these methods mainly focus on which modules to use for aggregating tokens rather than how to construct the tokens (about patchify stem and stage design). On the other hand, efficient MHSA techniques reduce the cost of attention by implementing sparse attention [9, 13–18] or low-rank approximation [19–22]. These modules are applied with the commonly adopted multi-head mechanism. Despite all the great progress, redundancies in macro/micro design are still not fully understood or addressed. *In this paper, we explore the redundancy at all design levels, and propose memory-efficient solutions.*

To identify computational redundancy in macro design, we concentrate on patch embedding size, observing that

*Corresponding author (E-mail: youngmin.ro@uos.ac.kr)

most recent efficient models use a 4×4 patch embedding. We conduct experiments as shown in Fig. 2 to analyze spatial redundancy in traditional macro design and find several intriguing results. First, despite having fewer channels, the early stages exhibit a severe speed bottleneck due to the large number of tokens (at 224×224 , stage1: 3136 tokens; stage2: 784 tokens). Second, using a 3-stage design that processes 196 tokens in the first stage through a 16×16 patchify stem does not lead to a significant drop in performance. For further comparison, we set up a basic model (Tab. 6 (2)) employing the aforementioned macro design, performing simple token mixing using a 3×3 depthwise convolution. Compared to the efficient model MobileViT-XS [18], our simple model achieves 1.5% superior accuracy on ImageNet-1k [23], while running $5.0 \times / 7.6 \times$ faster on the A100 GPU / Intel CPU. *These results demonstrate that there is considerable spatial redundancy in the early stages, and compared with specialized attention methods, efficient macro design is more crucial for the model to achieve competitive performance within strict latency limits.* Note that this observation does not mean the token mixer is trivial.

We also probe redundancy in micro design, specifically within the MHSA layer. Most efficient MHSA methods have primarily focused on effective spatial token mixing. Due to the efficient macro design, we are able to use compact token representations with increased semantic density. Thus, we turn our focus to the channel (head) redundancy present in attention layers, also crucial aspect overlooked in most previous works. Through comprehensive experiments, we find that there is a noticeable redundancy in multi-head mechanism, particularly in the latter stages. We then propose a novel Single-Head Self-Attention (SHSA) as a competitive alternative that reduces the computational redundancy. In SHSA, self-attention with a single head is applied to just a subset of the input channels, while the others remain unchanged. *SHSA layer not only eliminates the computational redundancy derived from multi-head mechanism but also reduces memory access cost by processing partial channels.* Also, these efficiencies enable stacking more blocks with a larger width, leading to performance improvement within the same computational budget.

Based on these findings, we introduce a Single-Head Vision Transformer (SHViT) based on memory-efficient design principles, as a new family of networks that run highly fast on diverse devices. Experiments demonstrate that our SHViT achieves state-of-the-art performance for classification, detection, and segmentation tasks in terms of both speed and accuracy, as shown in Fig. 1. For instance, our SHViT-S4 achieves 79.4% top-1 accuracy on ImageNet with throughput of 14283 images/s on an Nvidia A100 GPU and 509 images/s on an Intel Xeon Gold 5218R CPU @ 2.10GHz, outperforming EfficientNet-B0 [24] by 2.3% in accuracy, 69.4% in GPU inference speed, and 90.6% in

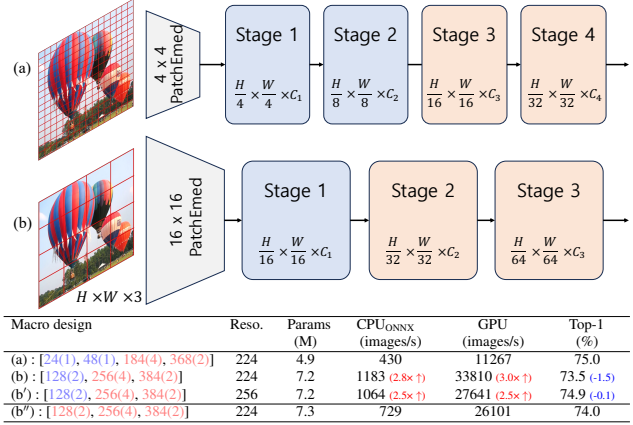


Figure 2. **Macro design analysis.** All stages are composed of MetaFormer blocks [28]. The stages depicted in blue and red utilize depthwise convolution and attention layers as token mixer, respectively. In the table below, the macro design numbers represent the number of channels, while the numbers in parentheses indicate the number of blocks.

CPU speed. Also, SHViT-S4 has 1.3% better accuracy than MobileViTv2 $\times 1.0$ [21] and is $2.4 \times$ faster on iPhone12 mobile device. For object detection and instance segmentation on MS COCO [25] using Mask-RCNN [26] detector, our model significantly outperforms EfficientViT-M4 [27] by 6.2 AP^{box} and 4.9 AP^{mask} with a smaller backbone latency on various devices.

In summary, our contributions are as follows:

- We conduct a systematic analysis of the redundancy that has been overlooked in the majority of existing research, and propose memory-efficient design principles to tackle it.
- We introduce Single-Head ViT (SHViT), which strike a good accuracy-speed tradeoff on a variety of devices such as GPU, CPU, and iPhone mobile device.
- We carry out extensive experiments on various tasks and validate the high speed and effectiveness of our SHViT.

2. Analysis and Method

In this section, we first conduct analyses of redundancies in both macro and micro design through first-of-its-kind experiments and then discuss various solutions to mitigate them. After that, we introduce the Single-Head Vision Transformer (SHViT) and explain its details.

2.1. Analysis of Redundancy in Macro Design

Most efficient models [5, 7–9, 11, 17, 19, 21, 29, 30] adopt a 4×4 patchify stem / 4-stage configuration (Fig. 2 (a)). In

	Query pixel	Most important head	Most redundant head	Head similarity (%)		Remove 1 head ($\Delta\%$)		Leave 1 head ($\Delta\%$)	
				3heads	6heads	3heads	6heads	3heads	6heads
DeiT Tiny (6heads): 72.14% (72.95%)									
(a) Layer 1				49.5	63.9	-0.81	-0.15	-1.71	-1.13
Layer 2				35.5	57.7	-0.95	-0.18	-2.60	-2.21
Layer 3				18.4	34.8	-1.51	-0.27	-4.50	-4.63
(b) Layer 4				39.2	33.8	-0.59	-0.25	-2.65	-4.51
Layer 5				53.2	46.2	-0.52	-0.28	-1.22	-2.40
Layer 6				47.5	39.7	-0.11	-0.31	-0.51	-2.13
Layer 7				52.1	41.3	-0.64	-0.37	-1.54	-1.61
Layer 8				63.5	53.2	-0.76	-0.05	-1.51	-1.67
Layer 9				69.0	64.2	-0.68	-0.28	-1.40	-1.35
(b) Layer 10				76.2	53.9	-0.76	-0.36	-1.17	-1.73
Layer 11				91.6	81.2	-0.55	-0.09	-0.92	-1.08
Layer 12				91.2	71.5	-0.26	-0.04	-0.54	-0.74

Figure 3. **Multi-head redundancy analysis on DeiT [31]**. To better analyze head redundancy, we increase the number of heads in DeiT-T from 3 to 6 and retrain the model. We compute the attention maps and calculate the average cosine similarity between each head in different layers across 128 test samples from ImageNet. The importance of each head is determined by its score when it is removed and when it is left alone. Zoom-in for better visibility.

contrast, plain ViT models [1, 31] adopt a 16×16 patchify stem to generate meaningful input tokens for subsequent MHSA layers. We focus on this discrepancy and further hypothesize that a larger-stride patchify stem is not only necessary for the MHSA layers but also crucial for effective representation learning within tight latency regimes.

To substantiate our hypothesis, inspired by [10, 27, 32], we adopt 16×16 patchify stem and 3-stage design. We build two models based on the MetaFormer block [28] and the two aforementioned macro designs (see Fig. 2 for details). Specifically, we configured both models to have a similar number of channels for equivalent feature map size. Surprisingly, model (b) is $3.0 \times / 2.8 \times$ faster on the GPU / CPU, respectively, although it performs 1.5% worse than (a). Furthermore, when trained at a resolution of 256×256 , (b') is not only comparable to (a) but also significantly faster.

As shown in the above observations, our proposed efficient macro design has the following advantages:

- 1) token representations with large receptive fields and reduced spatial redundancy can be utilized at the early stages.
- 2) It can diminish the feature map size by up to 16 times, leading to a significant reduction in memory access costs.
- 3) due to the aggressive stride design, there's only a mild decrease in throughput when the resolution is increased, leading to effective performance enhancement (as shown in Fig. 2(b'), 8, and Tab. 2).

2.2. Analysis of Redundancy in Micro Design

MHSA layer computes and applies attention maps independently in multiple subspaces (heads), which has consistently shown enhanced performance [1, 33]. However,

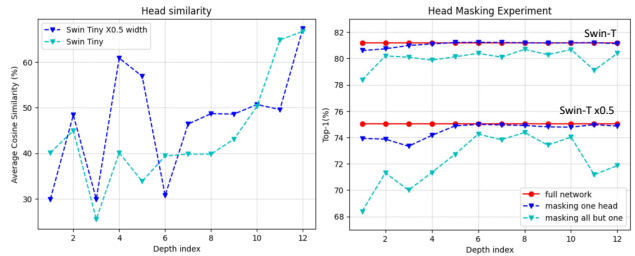


Figure 4. **Multi-head redundancy analysis on Swin [40]**. We scale down by halving the width of Swin-T. **Left**: the average cosine similarity. **Right**: head masking results. The process of deriving the results is the same as the DeiT experiment. (Fig. 3)

while attention maps are computationally demanding, recent studies have shown that many of them aren't critically essential [34–39]. We also delve into the multi-head redundancy of prevailing tiny ViT models (DeiT-T [31], Swin-T [40]) through three experiments: *attention map visualization*, *head similarity analysis*, and *head ablation study*.

For head similarity analysis, we measure the average cosine similarity between each head and other heads in the same layer. For head ablation study, we evaluate the performance impact by nullifying the output of some heads in a given layer while maintaining full heads in the other layers. And the highest score is reported. Details of each experiment and further results are provided in the supplementary materials.

First of all, in the early stages (Fig. 3 (a)), we observe that the top-performing heads tend to operate in a convolution-like manner, while heads that have minimal impact on performance when removed typically process information more globally. Also, as shown in Fig. 2 (b'), the model using attention layers in the first stage exhibits a less favorable speed-accuracy trade-off compared to those employing depthwise convolution layers in the first stage. Hence, for efficiency, we use convolutions with spatial inductive bias as the token mixer in the initial stage.

In the latter stages, we find that there is a lot of redundancy both at the feature and prediction levels. For example, the latter stages of DeiT-T (Fig. 3 (b)) exhibit the average head similarity of 78.3% (64.8% for 6 heads), with Swin-T also demonstrating notably high values (Fig. 4 Left). In the experiment of removing one head, we observe that the majority of heads can be removed without deviating too much from the original accuracy. Remarkably, in some cases of Swin-T (Fig. 4 Right), removing a head even leads to slightly improved score. Furthermore, when using just one head out of 12 or 24 in Swin-T, the performance drop is, on average, only 0.95% points.

Previous approaches [34–39] to tackle head redundancy typically train full networks first and then prune unnecessary heads. Although these methods are effective, they come at the expense of increased computational resources and

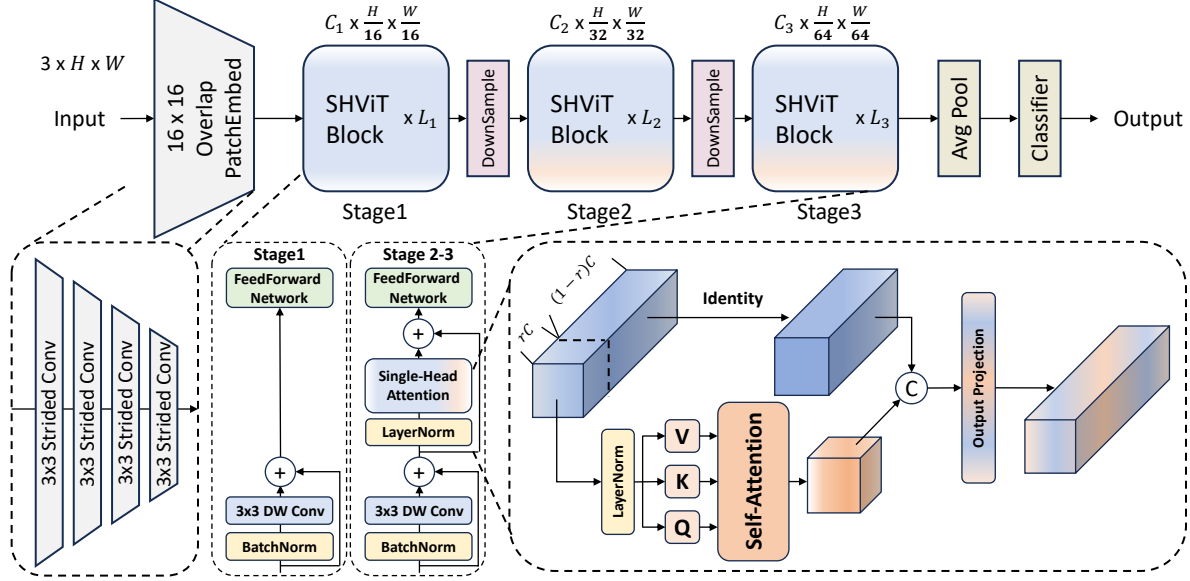


Figure 5. **Overview of Single-Head Vision Transformer (SHViT).** The model starts with a 16×16 overlapping patch embedding layer and uses single-head attention layers in the latter stages to efficiently compute global dependencies. See text for details.

memory footprints during training. To address the aforementioned problem cost-effectively, we design our attention module with a single head to inherently avoid head redundancy. This approach ensures both the training and inference processes are streamlined and efficient.

2.3. Single-Head Self-Attention

Based on the above analyses, we propose a new Single-Head Self-Attention (SHSA), with details presented in the lower right corner of Fig. 5. It simply applies an attention layer with a single head on only a part of the input channels ($C_p = rC$) for spatial feature aggregation and leaves the remaining channels untouched. We set r to $1/4.67$ as a default. Formally, the SHSA layer can be described as:

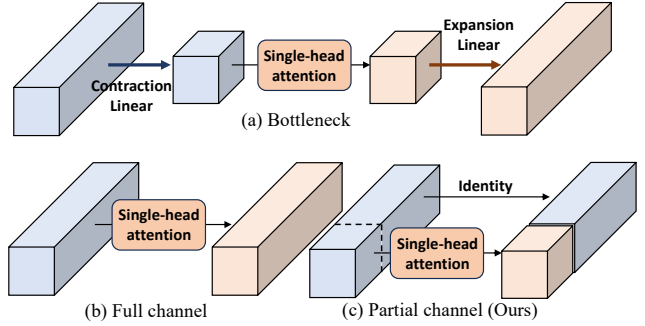
$$\text{SHSA}(\mathbf{X}) = \text{Concat}(\tilde{\mathbf{X}}_{att}, \mathbf{X}_{res})W^O \quad (1)$$

$$\tilde{\mathbf{X}}_{att} = \text{Attention}(\mathbf{X}_{att}W^Q, \mathbf{X}_{att}W^K, \mathbf{X}_{att}W^V), \quad (2)$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}(\mathbf{QK}^\top / \sqrt{d_{qk}})\mathbf{V}, \quad (3)$$

$$\mathbf{X}_{att}, \mathbf{X}_{res} = \text{Split}(\mathbf{X}, [C_p, C - C_p]) \quad (4)$$

where W^Q , W^K , W^V , and W^O are projection weights, d_{qk} is the dimension of the query and key (set to 16 as a default), and $\text{Concat}(\cdot)$ is the concatenation operation. For consistent memory access, we take the initial C_p channels as representatives of the whole feature maps. Additionally, the final projection of SHSA is applied to all channels, rather than just the initial C_p channels, ensuring efficient propagation of the attention features to the remaining channels. *SHSA can be interpreted as sequentially stretching the previously parallel-computed redundant heads along the block-axis.*



Single-Head Attention design	Params (M)	CPU _{ONNX} (images/s)	GPU (images/s)	Top-1 (%)
(a) : Bottleneck	10.6	802	27195	73.6
(b) : Full channel	10.4	724	26432	75.1
(c) : Partial channel (Ours)	11.4	951	26878	75.2

Figure 6. **Comparison of Single-head attention designs.** (a) replaces convolution with single-head attention in ResNet’s bottleneck block [4]. The contraction ratio is equal to the partial ratio in (c). (b) uses full channels for single-head attention modules. All models are configured to have similar speeds. Our partial channel approach has the best speed-accuracy tradeoff.

In Fig. 6, we also explore various single-head designs. Recent studies [7, 12, 27, 29, 32, 41, 42] sequentially combines convolution and attention layers to incorporate local details into a global contexts. Unfortunately, this approach can only extract either local detail or global context in a given token mixer. Also, it is noted in [6] that some channels process local details while others handle global modeling. These observations imply that the current serial approaches have redundancy when processing all channels in each layer

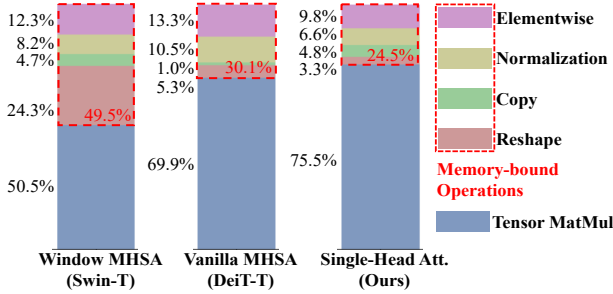


Figure 7. **Runtime breakdown.** Operations highlighted in the red box represent memory-bound operations, where the majority of the duration is consumed by memory accesses, and the computational time is relatively brief.

(Fig. 6 (a), (b)). In contrast, our partial channel approach with preceding convolution memory-efficiently addresses the aforementioned issue by leveraging two complementary features in parallel within a single token mixer [12, 43].

For effective utilization of the attention layer, Layer Normalization [44] is essential; meanwhile, to implement a multi-head approach, data movements like reshape operation are required. Consequently, as shown in Fig. 7, a large portion of MHSA’s runtime is taken up by memory-bound operations like reshaping and normalization [27, 45–47]. By minimizing the use of memory-bound operations or applying them to fewer input channels, the SHSA module can fully leverage the computing power of GPUs/CPUs.

2.4. Single-Head Vision Transformer

An overview of the Single-Head Vision Transformer (SHViT) architecture is illustrated in Fig. 5. Given an input image, we first apply four 3×3 strided convolution layers to it. Compared to the stride-16 16×16 convolution tokenizing used by standard ViT models [1, 31], our overlapping patchify stem can extract better local representations [10, 27, 32]. Then, the tokens pass through three stages of stacked SHViT blocks for hierarchical representation extraction. A SHViT block consists of three main modules (see Fig. 5): Depthwise Convolution (DWConv) layer for local feature aggregation or conditional position embedding [48, 49], Single-Head Self-Attention (SHSA) layer for modeling global contexts, and Feed-Forward-Network (FFN) for channel interaction. The expansion ratios in FFN are set to 2. The combination of DWConv and SHSA captures both local and global dependencies in a computationally and memory-efficient manner. Based on findings in sec. 2.2, we do not use the SHSA layer in the first stage. To reduce tokens without information loss, we utilize an efficient downsampling layer, which is composed of two stage 1 blocks, with an inverted residual block [27, 50, 51] (stride-2) placed between them. Finally, the global average pooling

Model variants	Depth	Emb. dim.	Reso.	Partial ratio	Exp. ratio
SHViT-S1	[2, 4, 5]	[128, 224, 320]	224	1 / 4.67	2
SHViT-S2	[2, 4, 5]	[128, 308, 448]	224		
SHViT-S3	[3, 5, 5]	[192, 352, 448]	224		
SHViT-S4	[4, 7, 6]	[224, 336, 448]	256		

Table 1. Architecture details of SHViT variants.

and fully connected layer are used to output the predictions.

Besides the aforementioned operators, normalization and activation layers also play crucial roles in determining the model speed. We employ Layer Normalization [44] only for the SHSA layer while integrating Batch Normalization (BN) [52] into the remaining layers, as BN can be merged into its adjacent convolution or linear layers. We also use ReLU [53] activations instead of other complex alternatives [51, 54, 55], as they are much slower on various inference deployment platforms [7, 27, 56].

We build four SHViT variants with different settings of depth and width. Due to the large-sized patch embedding and single-head design, we can use a larger number of channels and blocks than previous efficient models. Model specifications are provided in Tab. 1.

3. Experiments

3.1. Implementation Details

We conduct image classification on ImageNet-1K [23], which includes 1.28M training and 50K validation images for 1000 categories. All models are trained from scratch using AdamW [57] optimizer for 300 epochs with a learning rate of 10^{-3} and a total batch size of 2048. We use cosine learning rate scheduler [58] with linear warmup for 5 epochs. Weight decays are set to 0.025/0.032/0.035/0.03 for SHViT-S1 to S4. For fair comparison, we follow the same data augmentation proposed in [31], including Mixup [59], random erasing [60], and auto-augmentation [61]. For 384^2 and 512^2 resolution, we finetune the model for 30 epochs with weight decay of 10^{-8} and learning rate of 0.004. Additionally, we assess throughput performance across various hardware platforms. We measure GPU throughput on an Nvidia A100 with batch size of 256. For CPU and CPU_{ONNX}, we evaluate the runtime on an Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz processor, with batch size of 16 (using a single thread). For CPU_{ONNX}, we convert the models to ONNX [62] runtime format. Mobile latency is measured using iPhone 12 with iOS version 16.5. We export the models (batch size is set to 1) using CoreML tools [63] and report the median latency over 1,000 runs. We also validate our model as an efficient vision backbone for object detection and instance segmentation on COCO [25] with RetinaNet [64] and Mask R-CNN [26], respectively. All models are trained under $1 \times$ schedule (12 epochs) following [40] on mmdetection library [65].

Model	Reso.	Epochs	FLOPs (M)	Params (M)	Throughput (images/s)			Top-1 (%)	Top-5 (%)
					GPU	CPU	CPU _{ONNX}		
MobileNetV3-Small [51]	224	600	57	2.5	31477	167	1172	67.4	-
MobileViT-XXS [18]	256	300	410	1.3	7594	21	170	69.0	-
MobileViTV2 ×0.5 [21]	256	300	466	1.4	8616	17	157	70.2	-
EfficientViT-M2 [27]	224	300	201	4.2	30377	147	781	70.8	90.2
MobileOne-S0 [56]	224	300	275	2.1	19689	86	1648	71.4	-
EMO-1M [29]	224	300	261	1.3	10032	34	119	71.5	-
FasterNet-T0 [30]	224	300	340	3.9	23518	92	844	71.9	-
ShuffleNetV2 ×1.5 [66]	224	300	299	3.5	16495	62	799	72.6	-
MobileFormer-96M [19]	224	450	96	3.6	13106	91	235	72.8	-
SHViT-S1	224	300	241	6.3	33489	143	1111	72.8	91.0
EfficientFormerV2-S0 [9]	224	300	400	3.5	2374	54	372	73.7	-
EfficientViT-M4 [27]	224	300	299	8.8	26201	113	616	74.3	91.8
EdgeViT-XXS [17]	224	300	600	4.1	6763	33	168	74.4	-
MobileViT-XS [18]	256	300	986	2.3	4408	8	96	74.8	-
ShuffleNetV2 ×2.0 [66]	224	300	591	7.4	12276	40	250	74.9	92.4
EMO-2M [29]	224	300	439	2.3	7333	25	78	75.1	-
MobileNetV3-Large [51]	224	600	217	5.4	13994	43	613	75.2	-
SHViT-S2	224	300	366	11.4	26878	99	951	75.2	92.4
FastViT-T8 [7]	256	300	700	2.1	5978	23	140	75.6	-
GhostNet ×1.3 [67]	224	300	226	7.3	9433	39	109	75.7	92.7
FasterNet-T1 [30]	224	300	850	7.6	17827	41	552	76.2	-
EfficientNet-B0 [24]	224	350	390	5.3	8433	26	267	77.1	93.3
EfficientViT-M5 [27]	224	300	522	12.4	18722	64	456	77.1	93.4
PoolFormer-S12 [28]	224	300	1823	11.9	5432	13	120	77.2	-
MobileOne-S2 [56]	224	300	1299	7.8	9355	22	581	77.4	-
SHViT-S3	224	300	601	14.2	20522	62	731	77.4	93.4
EdgeViT-XS [17]	224	300	1100	6.7	5520	21	120	77.5	-
EfficientFormerV2-S1 [9]	224	300	650	6.1	2112	37	325	77.9	-
MobileViTV2 ×1.0 [21]	256	300	1800	4.9	4345	7	63	78.1	-
ResNet50 [4, 5]	224	300	4110	25.6	5281	8	271	78.8	-
FasterNet-T2 [30]	224	300	1910	15.0	11181	21	417	78.9	-
EMO-6M [29]	224	300	961	6.1	5105	15	50	79.0	-
EfficientNet-B1 [24]	240	350	700	7.8	4982	11	156	79.1	94.4
FastViT-T12 [7]	256	300	1400	6.8	4197	14	92	79.1	-
MobileFormer-508M [19]	224	450	508	14.0	5390	23	91	79.3	-
MobileOne-S4 [56]	224	300	2978	14.8	5281	11	281	79.4	-
SHViT-S4	256	300	986	16.5	14283	36	509	79.4	94.5
Finetuning with higher resolution									
EfficientViT-M5 _{r=384} [27]	384	330	1486	12.4	7041	17	176	79.8	95.0
EfficientViT-M5 _{r=512} [27]	512	360	2670	12.4	3777	9	88	80.8	95.5
SHViT-S4 _{r=384}	384	330	2225	16.5	6702	14	315	81.0	95.4
SHViT-S4 _{r=512}	512	360	3973	16.5	3957	8	198	82.0	95.9

Table 2. SHViT classification performance on ImageNet-1K [23] with comparisons to SOTA efficient models. Throughput is measured on an Nvidia A100 GPU with batch size of 256 for GPU and Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz processor with batch size of 16 for CPU and CPU_{ONNX}. Larger throughput means faster inference speed. FLOP count is computed by fvcore [68] library.

3.2. SHViT on ImageNet-1K Classification

As shown in Fig. 1, Tab. 2, and 4, we compare Single-Head Vision transformer (SHViT) with the state-of-the-art models. The comparison results clearly show that our SHViT achieves a better trade-off between accuracy and throughput/latency across various devices.

Comparison with efficient CNNs. Our SHViT-S1 achieves 5.4% higher accuracy than MobileNetV3-Small [51], while maintaining similar speeds on both the A100 GPU and Intel CPU. Compared to ShuffleNetV2 ×2.0 [66], SHViT-S2 obtains slightly better performance with 2.2× and 2.5× speed improvements on the A100 GPU and Intel CPU, respectively. Furthermore, our model is 3.8× faster when converted to the ONNX runtime format. Compared to the recent FasterNet-T1 [30], SHViT-S3 not only achieves 1.2%

higher accuracy but also runs faster: 15.1% on the A100 GPU and 32.4% on the Intel CPU. Notably, at Top-1 accuracy of 79.1–79.4%, our model is 2.9×/ 3.3× faster than EfficientNet-B1 [24], 3.4×/ 5.5× faster than FastViT-T12 [7], and 2.7×/ 1.8× faster than MobileOne-S4 [56] on the A100 GPU/Intel CPU with ONNX format. *When leveraging minimal attention module in a memory-efficient way, ViTs can still show fast inference speeds like efficient CNNs.*

Comparison with efficient ViTs and hybrid models.

Our SHViT-S1 has 10% and 42% higher throughput than EfficientViT-M2 [27] on the A100 GPU and Intel CPU with ONNX runtime, respectively, while showing better performance with a large margin (70.8% → 72.8%). SHViT-S3 obtains similar accuracy to PoolFormer-S12 [28], but it uses 3× fewer FLOPs, is 3.8× faster on the A100 GPU, and

Model	Reso.	Flops (M)	Throughput (images/s)			Top-1 (%)
			GPU	CPU	CPU _{ONNX}	
SHViT-S1	224	241	33489	143	1111	74.0
SwiftFormer-XS [69]	224	600	7922	26	175	75.7
EfficientFormerV2-S0 [9]	224	400	2374	54	372	75.7
SHViT-S2	224	366	26878	99	951	76.2
FastViT-T8 [7]	256	1400	5978	23	140	76.7
SHViT-S3	224	601	20522	62	731	78.3
SwiftFormer-S [69]	224	1000	6415	21	147	78.5
EfficientFormerV2-S1 [9]	224	650	2112	37	325	79.0
EfficientFormer-L1 [8]	224	1300	6840	21	274	79.2
SHViT-S4	256	986	14283	36	509	80.2

Table 3. Comparison of SOTA efficient models on ImageNet-1K classification, using DeiT [31] distillation recipe.

6.1× faster as ONNX model. Remarkably, our SHViT-S4 surpasses recent EdgeViT-XS [17] with a 1.9% higher accuracy, while being 2.6× faster on A100 GPU, 1.7× faster on Intel CPU, and 4.2× faster on ONNX implementation. As shown in the results above, when converted to ONNX format, our models demonstrate a notable performance boost compared to the recent SOTA models. This enhancement is largely because our single-head design uses fewer reshape operations, which often cause overhead in ONNX runtime. *To summarize, the above results demonstrate that our proposed memory-efficient macro design has a more significant impact on the speed-accuracy tradeoff than efficient attention variants or highly simple operations like pooling.*

Finetuning with higher resolution. Following [27], we also finetune our SHViT-S4 to higher resolutions. Compared to the state-of-the-art EfficientViT-M5_{r512} [27], our SHViT-S4_{r384} attains competitive performance, even when trained at a lower resolution. Additionally, SHViT-S4_{r384} is 77.4% faster on the A100 GPU, 55.6% on the Intel CPU, and an impressive 3.6× faster on ONNX runtime format. Moreover, SHViT-S4_{r512} achieves 82.0% top-1 accuracy with throughput of 3957 images/s on the A100 GPU, demonstrating effectiveness across various input sizes.

Distillation results. We report the performance of our models using DeiT [31] distillation recipe in Tab. 3. Notably, our models outperform competing models in both speed and accuracy. Specifically, SHViT-S3 even surpasses FastViT-T8 which is 5.2× slower as ONNX models. SHViT-S4 attains superior performance than EfficientFormer-L1 [8] while being 2.1× / 1.9× faster on the GPU / ONNX runtime.

Model	Latency (ms)	Top-1 (%)
EfficientFormer-L1 [8]	1.5	77.3 (79.2)
EfficientFormerV2-S1 [9]	1.3	77.9 (79.0)
MobileViTv2 × 1.0 [21]	3.8	78.1
EfficientNet-B1 [24]	1.8	79.1
FastViT-T12 [7]	1.4	79.1 (80.3)
MobileOne-S4 [56]	1.7	79.4
SHViT-S4	1.6	79.4 (80.2)

Table 4. Mobile latency comparison. The results in brackets are trained with distillation [31].

faster on iPhone 12 device. SHViT-S4 also obtains com-

Mobile Latency Evaluation. We also verify the effectiveness of our model on the mobile device in Tab. 4. Compared to the efficient models EfficientNet-B1 [24] / MobileOne-S4 [56], our SHViT-S4 achieves similar accuracy while running 0.2 ms / 0.1 ms

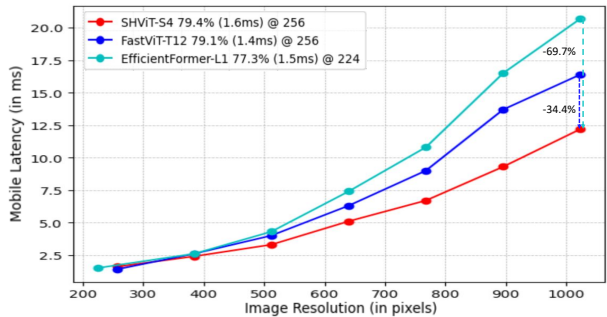


Figure 8. Mobile latency comparison of a SHViT-S4 with recent state-of-the-art FastViT [7] and EfficientFormer [8]; measured on iPhone 12 for various input sizes.

RetinaNet Object Detection on COCO									
Backbone	Latency (ms)			AP	AP ₅₀	AP ₇₅	AP _s	AP _m	AP _l
	GPU	CPU	Mobile						
MobileNetV3 [51]	0.34	7.5	7.5	29.9	49.3	30.8	14.9	33.3	41.1
EfficientViT-M4 [27]	0.33	7.3	7.8	32.7	52.2	34.1	17.6	35.3	46.0
PVTv2-B0 [70]	0.73	115.4	27.5	37.2	57.2	39.5	23.1	40.4	49.7
MobileFormer-508M [19]	0.89	35.7	26.9	38.0	58.3	40.3	22.9	41.2	49.7
EdgeViT-XXS [17]	0.88	38.4	12.9	38.7	59.0	41.0	22.4	42.0	51.6
SHViT-S4	0.28	5.0	3.3	38.8	59.8	41.1	22.0	42.4	52.7

Mask R-CNN Object Detection & Instance Segmentation on COCO									
Backbone	Latency (ms)			AP ^b	AP ^b ₅₀	AP ^b ₇₅	AP ^m	AP ^m ₅₀	AP ^m ₇₅
	GPU	CPU	Mobile						
EfficientNet-B0 [24]	0.54	16.7	3.8	31.9	51.0	34.5	29.4	47.9	31.2
EfficientViT-M4 [27]	0.33	7.3	7.8	32.8	54.4	34.5	31.0	51.2	32.2
PoolFormer-S12 [28]	1.20	40.4	6.8	37.3	59.0	40.1	34.6	55.8	36.9
EfficientFormer-L1 [8]	0.84	21.0	4.3	37.9	60.3	41.0	35.4	57.3	37.3
ResNet-50 [4]	0.94	19.0	8.8	38.0	58.6	41.4	34.4	55.1	36.7
FastViT-SA12 [7]	1.06	39.4	6.5	38.9	60.5	42.2	35.9	57.6	38.1
SHViT-S4	0.28	5.0	3.3	39.0	61.2	41.9	35.9	57.9	37.9

Table 5. Comparison results on object detection and instance segmentation on COCO 2017 [25] using RetinaNet [64] and Mask RCNN [26] head. Backbone latencies are measured with image crops of 512 × 512. The batch sizes used for GPU, CPU, and Mobile latency are 32, 16, and 1 respectively.

petitive performance against highly-optimized models for mobile latency, indicating its consistent performance across diverse inference platforms. Further results in Fig. 8 show that our model significantly outperforms over the recent models FastViT [7] and EfficientFormer [8], especially at higher resolutions. At low resolutions, SHViT-S4 is slightly slower, but at 1024 × 1024, our model achieves 34.4% and 69.7% lower latency than FastViT and EfficientFormer, respectively. These results stem from the increased memory efficiency in the macro and micro design.

3.3. SHViT on downstream tasks

In Tab. 5, We evaluate the transfer ability of our SHViT using two frameworks: 1) RetinaNet [64] for object detection, 2) Mask R-CNN [26] for instance segmentation.

Object detection. SHViT-S4 is 2.3× faster on mobile device than MobileNetV3 [51] and outperforms it by +8.9 AP. Compared to MobileFormer [19], our model achieves better performance while being 3.2× and 8.2× faster on the A100 GPU and mobile device, respectively.

#Row	Ablation	Variant	Throughput (images/s)			Top-1 (%)
			GPU	CPU	CPU _{ONNX}	
(1)	Single Head	→ MHSA [1, 33]	18036	50	578	77.7
(2)	Self-Attention	→ None	22075	61	792	76.3
(3)		= 1/8	20666	57	754	77.1
(4)	Partial Ratio	= 1/4.67 (SHViT-S3)	20522	62	731	77.4
(5)		= 1/2	19976	56	673	77.5

Table 6. Ablation on our proposed Single-Head Attention and design choice for SHViT-S3 variant.

Instance Segmentation. SHViT-S4 surpasses GPU or mobile-optimized models like EfficientViT [27] and EfficientNet [24] in speed, while delivering a substantial performance boost. Remarkably, our model gains 1.7 AP^b and 1.3 AP^m over PoolFormer [28] but runs 4.3×, 8.1×, and 2.1× faster on the GPU, CPU, and mobile device respectively.

As shown in the above results, the large-stride patchify stem with 3-stage reduces not only computational costs but also generates meaningful token representations, especially at higher resolutions. Furthermore, the marked performance gap with EfficientViT [27], using a similar macro design, proves the efficacy of our micro design choices.

3.4. Ablation Study

In this section, we first verify the effectiveness of our proposed Single-Head Self-Attention (SHSA) layer and then conduct a concise ablation study on the value of the partial ratio for SHSA layer. Results are provided in Tab. 6.

Effectiveness of SHSA. To assess whether the SHSA layer can effectively capture the global contexts like the Multi-Head Self-Attention (MHSA) [1] layer, we conduct an ablation study by either replacing the SHSA layer with the MHSA layer or removing it. As shown in Tab. 6 (1, 2 vs. 4), SHSA layer exhibits a better speed-accuracy tradeoff compared to MHSA layer. While removing SHSA layer results in a faster model speed, it leads to a significant drop in accuracy. Meanwhile, model (2) can also achieve highly competitive performance compared with the SOTA models in Tab. 2, which shows that our proposed macro design offers a solid architectural baseline under tight latency constraints.

Searching for the appropriate partial ratio of SHSA. By default, we set the partial ratio to 1 / 4.67 for all SHViT models, which obtains the optimal speed-accuracy tradeoff (3, 5 vs. 4). Compared to a very small value, increasing the channels moderately for token interaction achieves effective performance enhancement at low costs. Also, a too large value does not provide a performance boost that compensates for the accompanying costs.

4. Related Work

Leveraging Convolutional Neural Networks (CNN) in resource-constrained devices has gained significant attention from many researchers. Within this trend, several strategies have emerged, including decomposition of convolution in MobileNets [50, 51, 71], channel shuffling in

ShuffleNets [66, 72], cheap linear transformation in GhostNet [67], compound scaling law in EfficientNet [24], and structural re-parameterization in many works [7, 56, 73].

Even within the Vision Transformer (ViT) [1] realm, there are ongoing numerous efforts for efficient designs to accelerate inference speed on various devices. One promising approach is designing a new ViT architecture that integrates the local priors of CNN. This method mostly incorporates attention only in the latter stages, allowing for the efficient extraction of global information without considerable computational overhead [7–10, 12]. In contrast, other methods employ attention and convolution in parallel, either within a single token mixer [14, 43, 74] or on a block-by-block basis [19], to combine a rich set of features. Another line of approach focuses on reducing the computational complexity of MHSA [17, 18, 21, 69]. For example, MobileViTv2 [21] introduces a separable self-attention with linear complexity with respect to the number of tokens (resolution). EdgeViT [17] applies MHSA to sub-sampled features to perform approximately full spatial interaction in a cost-effective manner. Unlike the above approaches, we prioritize organizing tokens with minimal spatial redundancy over efficiently mixing tokens.

Also, recent works [27, 34–39, 75–77] have demonstrated that numerous heads function in similar ways and can be pruned without notably affecting performance. EfficientViT [27] proposes feeding attention heads with different splits of the full channel to improve attention diversity. In addition, [76] presents a regularization loss for multi-head similarity, while [78] explores head similarity across different layers. As opposed to reducing multi-head redundancy, we design module with single-head configuration, which not only inherently prevents multi-head redundancy but also saves computation costs.

5. Conclusion

In this work, we have investigated redundancies at both the spatial and channel dimensions of the architectural design commonly used by many established models. We then proposed 16×16 patch embeddings with 3-scale hierarchical representations and Single-Head Self-Attention to address the computational redundancies. We further present our versatile SHViT, built upon our proposed macro/micro designs, that achieves ultra-fast inference speed and high performance on diverse devices and vision tasks.

6. Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NO.RS-2022-00166109) and (2022M3J6A1084845).

References

- [1] 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](#), [3](#), [5](#), [8](#)
- [2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020. [1](#)
- [3] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34:17864–17875, 2021. [1](#)
- [4] 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](#), [4](#), [6](#), [7](#)
- [5] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. [1](#), [2](#), [6](#)
- [6] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34:12116–12128, 2021. [1](#), [4](#)
- [7] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Fastvit: A fast hybrid vision transformer using structural reparameterization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. [1](#), [2](#), [4](#), [5](#), [6](#), [7](#), [8](#)
- [8] Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. *Advances in Neural Information Processing Systems*, 35:12934–12949, 2022. [1](#), [2](#), [7](#), [8](#)
- [9] Yanyu Li, Ju Hu, Yang Wen, Georgios Evangelidis, Kamyar Salahi, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Rethinking vision transformers for mobilenet size and speed. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16889–16900, October 2023. [1](#), [2](#), [6](#), [7](#), [8](#)
- [10] Benjamin Graham, Alaeldin 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12259–12269, 2021. [1](#), [3](#), [5](#), [8](#)
- [11] 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 *Computer Vision—ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VII*, pages 3–20. Springer, 2023. [1](#), [2](#)
- [12] Namuk Park and Songkuk Kim. How do vision transformers work? In *International Conference on Learning Representations*, 2022. [1](#), [4](#), [5](#), [8](#)
- [13] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. [1](#)
- [14] Zizheng Pan, Jianfei Cai, and Bohan Zhuang. Fast vision transformers with hilo attention. *Advances in Neural Information Processing Systems*, 35:14541–14554, 2022. [1](#), [8](#)
- [15] Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. Combiner: Full attention transformer with sparse computation cost. *Advances in Neural Information Processing Systems*, 34:22470–22482, 2021. [1](#)
- [16] 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](#)
- [17] 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 *Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI*, pages 294–311. Springer, 2022. [1](#), [2](#), [6](#), [7](#), [8](#)
- [18] Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In *International Conference on Learning Representations*, 2022. [1](#), [2](#), [6](#), [8](#)
- [19] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobileformer: Bridging mobilenet and transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5270–5279, 2022. [1](#), [2](#), [6](#), [7](#), [8](#)
- [20] Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. [1](#)
- [21] Sachin Mehta and Mohammad Rastegari. Separable self-attention for mobile vision transformers. *arXiv preprint arXiv:2206.02680*, 2022. [1](#), [2](#), [6](#), [7](#), [8](#)
- [22] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. [1](#)

- [23] 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. 2, 5, 6
- [24] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 2, 6, 7, 8
- [25] 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 *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 2, 5, 7
- [26] 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. 2, 5, 7
- [27] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. Efficientvit: Memory efficient vision transformer with cascaded group attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14420–14430, 2023. 2, 3, 4, 5, 6, 7, 8
- [28] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10819–10829, 2022. 2, 3, 6, 7, 8
- [29] Jiangning Zhang, Xiangtai Li, Jian Li, Liang Liu, Zhucun Xue, Boshen Zhang, Zhengkai Jiang, Tianxin Huang, Yabiao Wang, and Chengjie Wang. Rethinking mobile block for efficient attention-based models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1389–1400, 2023. 2, 4, 6
- [30] Jierun Chen, Shiu-hong Kao, Hao He, Weipeng Zhuo, Song Wen, Chul-Ho Lee, and S.-H. Gary Chan. Run, don’t walk: Chasing higher flops for faster neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12021–12031, June 2023. 2, 6
- [31] 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. 3, 5, 7
- [32] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. *Advances in Neural Information Processing Systems*, 34:30392–30400, 2021. 3, 4, 5
- [33] 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. 3, 8
- [34] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019. 3, 8
- [35] 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. *arXiv preprint arXiv:1905.09418*, 2019. 3, 8
- [36] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34:19974–19988, 2021. 3, 8
- [37] Zhuoran Song, Yihong Xu, Zhezhi He, Li Jiang, Naifeng Jing, and Xiaoyao Liang. Cp-vit: Cascade vision transformer pruning via progressive sparsity prediction. *arXiv preprint arXiv:2203.04570*, 2022. 3, 8
- [38] Zejiang Hou and Sun-Yuan Kung. Multi-dimensional model compression of vision transformer. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 01–06. IEEE, 2022. 3, 8
- [39] Huanrui Yang, Hongxu Yin, Maying Shen, Pavlo Molchanov, Hai Li, and Jan Kautz. Global vision transformer pruning with hessian-aware saliency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18547–18557, 2023. 3, 8
- [40] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 3, 5
- [41] 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. 4
- [42] Jianyuan Guo, Kai Han, Han Wu, Yehui Tang, Xinghao Chen, Yunhe Wang, and Chang Xu. Cmt: Convolutional neural networks meet vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12175–12185, 2022. 4
- [43] Chenyang Si, Weihao Yu, Pan Zhou, Yichen Zhou, Xinchao Wang, and Shuicheng YAN. Inception transformer. In *Advances in Neural Information Processing Systems*, 2022. 5, 8
- [44] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5
- [45] Andrei Ivanov, Nikoli Dryden, Tal Ben-Nun, Shigang Li, and Torsten Hoefler. Data movement is all you need: A case study on optimizing transformers. *Proceedings of Machine Learning and Systems*, 3:711–732, 2021. 5
- [46] Hamid Tabani, Ajay Balasubramaniam, Shabbir Marzban, Elahe Arani, and Bahram Zonooz. Improving the efficiency of transformers for resource-constrained devices. In *2021 24th Euromicro Conference on Digital System Design (DSD)*, pages 449–456. IEEE, 2021. 5

- [47] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022. 5
- [48] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, and Chunhua Shen. Conditional positional encodings for vision transformers. In *ICLR 2023*, 2023. 5
- [49] 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
- [50] 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. 5, 8
- [51] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019. 5, 6, 7, 8
- [52] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. 5
- [53] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 5
- [54] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 5
- [55] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic relu. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX 16*, pages 351–367. Springer, 2020. 5
- [56] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Mobileone: An improved one millisecond mobile backbone. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7907–7917, 2023. 5, 6, 7, 8
- [57] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5
- [58] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 5
- [59] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 5
- [60] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020. 5
- [61] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 113–123, 2019. 5
- [62] Ke Zhang et al Junjie Bai, Fang Lu. Onnx: Open standard for machine learning interoperability. <https://github.com/onnx/onnx>, 2019. 5
- [63] Core ml tools. <https://coremltools.readme.io/docs>, 2017. Use Core ML Tools to convert models from third-party libraries to Core ML. 5
- [64] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 5, 7
- [65] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 5
- [66] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018. 6, 8
- [67] 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. 6, 8
- [68] fvcore. fair. <https://github.com/facebookresearch/fvcore>, 2019. 6
- [69] Abdelrahman Shaker, Muhammad Maaz, Hanoona Rasheed, Salman Khan, Ming-Hsuan Yang, and Fahad Shahbaz Khan. Swiftformer: Efficient additive attention for transformer-based real-time mobile vision applications. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17425–17436, October 2023. 7, 8
- [70] 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. 7
- [71] 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. 8
- [72] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018. 8

- [73] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13733–13742, 2021. 8
- [74] Yufei Xu, Qiming Zhang, Jing Zhang, and Dacheng Tao. Vi-tae: Vision transformer advanced by exploring intrinsic inductive bias. *Advances in Neural Information Processing Systems*, 34:28522–28535, 2021. 8
- [75] Liyuan Liu, Jialu Liu, and Jiawei Han. Multi-head or single-head? an empirical comparison for transformer training. *arXiv preprint arXiv:2106.09650*, 2021. 8
- [76] Tianlong Chen, Zhenyu Zhang, Yu Cheng, Ahmed Awadallah, and Zhangyang Wang. The principle of diversity: Training stronger vision transformers calls for reducing all levels of redundancy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12020–12030, 2022. 8
- [77] zhuofan xia, Xuran Pan, Xuan Jin, Yuan He, Hui Xue', Shiji Song, and Gao Huang. Budgeted training for vision transformer. In *The Eleventh International Conference on Learning Representations*, 2023. 8
- [78] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021. 8