FreeNet: Liberating Depth-Wise Separable Operations for Building Faster Mobile Vision Architectures

Hao Yu[1], **Haoyu Chen**[1], **Wei Peng**[2], **Xu Cheng**[3], **Guoying Zhao**[1,4*]
[1]Center for Machine Vision and Signal Analysis, University of Oulu, Finland
[2]Department of Psychiatry and Behavioral Sciences, Stanford University, USA
[3]School of Computer Science, Nanjing University of Information Science and Technology, China
[4]Department of Computer Science, Aalto University, Finland
{hao.2.yu, chen.haoyu, guoying.zhao}@oulu.fi, wepeng@stanford.edu, xcheng@nuist.edu.cn

Abstract

In the pursuit of efficient vision architectures, substantial efforts have been devoted to optimizing operator efficiency. Depth-wise separable operators, such as DWConv, are found cheap in both FLOPs and parameters. As a result, they are increasingly incorporated into efficient backbones, trading for deeper and wider architectures to enhance performance. However, separable operators are not really fast on devices due to the discontinuous memory access requirements. In this paper, we propose **FreeNets**, a family of simple and efficient backbones that free the separable operation to further accelerate the running speed. We introduce sparse sampling mixers (S2-Mixer) to supersede existing separable token mixers. The S2-Mixer samples multiple segments of partially continuous signals across spatial and channel dimensions for convolutional processing, achieving extremely fast on-device speed. The sparse sampling also enables S2-Mixer to capture long-range pixel relationships from dynamic receptive fields. Furthermore, we introduce a Shift Feed-Forward Network (ShiftFFN) as a faster alternative to existing channel mixers. It utilizes a shift neck architecture that aggregates global information to shift features, enabling faster channel mixing while incorporating global pixel information. Extensive experiments demonstrate that FreeNet offers a superior accuracy-efficiency tradeoff compared to the latest efficient models. On ImageNet-1k, FreeNet-S2 outperforms the StarNet-S4 by 0.4% in top-1 accuracy, while running around 40% faster on desktop GPU and 15% faster on Mobile GPU.

Introduction

Designing efficient backbones has always been one of the research hotspots in computer vision. Recently, the topic of efficient backbones has been quickly offset to the mining of efficient Vision Transformers (ViT) (Chu et al. 2021; Li et al. 2022b; Shaker et al. 2023; Yun and Ro 2024). Owing to the powerful self-attention (SA) mechanism, ViTs demonstrate a superior ability to capture long-range pixel dependencies and aggregate pixels across diverse receptive fields. However, their quadratic complexity (Han et al. 2022) and the high dimensionality of image data have spurred extensive research endeavors dedicated to devising efficient ViTs that strike a favorable balance between accuracy and computational complexity. In the original ViT (Dosovitskiy et al. 2020), images are treated as 1D sequences of visual tokens. This approach, however, is inefficient as notable parameters are wasted in learning the visual inductive biases (e.g., locality and translation invariance) that are inherent in convolutions (Convs). Consequently, there's increasing interest in blending convolutional techniques and designs (Wang et al. 2021; Liu et al. 2021), like convolutional local representations, with ViTs to inject visual biases, making such hybrid models the forefront of vision backbone innovation.

Notably, like in prior lightweight CNNs, depth-wise separable convolution (DWConv) (Chollet 2017) has also gained prominence in recent ViTs (Vasu et al. 2023a; Shaker et al. 2023) for enhancing efficiency. The DWConv, by running on individual channels only, achieves a lower parameter count than vanilla Convs, which enables the expansion of network depth and width to improve performance. However, **its actual speed often falls short of expectations due to inefficient memory access** (Ma et al. 2018; Lu, Zhang, and Wang 2021), a problem also encountered with the linear SA (Wang et al. 2022). This inefficiency is largely due to its segmented processing kernels. Thus, despite the high theoretical efficiency of DWConv, its on-device speed lags significantly behind that of vanilla Convs. For example, with similar parameters, the DWConv-based ConvNeXt-T (Liu et al. 2022) operates nearly twice slower than ResNet-50 (He et al. 2016).

This paper challenges the conventional reliance on separable operations in efficient architectures, aiming to boost speed by eliminating such design stereotypes. To achieve this, we propose Sparse-wise Convolution (SWConv), which operates on a continuous subset of signals in a feature map, balancing speed and lightweight design. By skipping nearby pixels while preserving channel-level signal continuity, SWConv efficiently captures long-range pixel dependencies at a lower computational cost. Building on this, we introduce the sparse-sampling mixer (S2-Mixer), a novel token mixer that surpasses DWConv in speed while rivaling self-attention (SA) in modeling performance. Specifically, S2-Mixer employs multiple SWConvs to sample distinct feature subsets, capturing diverse spatial patterns across varying receptive fields. As shown in Figure 1(a), S2-Mixer processes images with dynamic receptive fields and achieves long-range pixel modeling akin to SA, yet with reduced run-

*Corresponding author

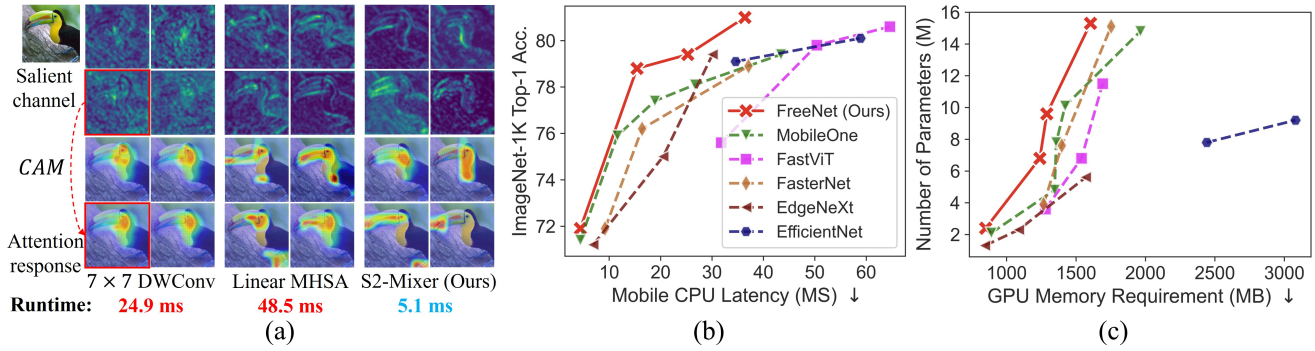

Figure 1: (a) Visualization of features captured by 7 × 7 DWConv, Linear MHSA (Wang et al. 2022), and our S2-Mixer. We present the top-4 salient channels and their attention maps (Selvaraju et al. 2017), respectively. (b) Comparison of ImageNet-1k Top-1 accuracy vs. mobile latency. (c) Comparison of CUDA GPU memory requirements vs. the number of model parameters.

time compared to DWConv. Additionally, S2-Mixer incorporates feature-level sparse operations, retaining partial signals from the previous layer to create an in-feature residual. Unlike existing token-channel mixer architectures (Yu et al. 2022; Wu et al. 2022), which require two residual branches to keep information flow, our framework eliminates the residual branch after the token mixer. This further improves speed and conserves memory.

Except for token-mixers, the Feed-Forward Network (FFN) is also employed in modern CNNs and ViTs as the channel mixer. It employs two PWConv (Chollet 2017) or linear layers to perform channel projection ($c \rightarrow 4c \rightarrow c$). However, this inverted residual scheme (Liu et al. 2022) is slow due to the increased channel width ($c \rightarrow 4c$) which causes excessive memory access. While recent studies have focused on accelerating token mixers, the FFN has gradually become the bottleneck, yet it remains underexplored. Additionally, to enhance performance, recent works (Wang et al. 2022; Shaker et al. 2023) tend to add DWConvs within the FFN for enhancing global spatial perception, which causes further deceleration. To this end, we propose a faster alternative called the Shift Feed-Forward Network (ShiftFFN). It reduces the channel expansion ratio of the first projection layer to reduce the memory access requirement. To remedy the loss in peak channel width, we design a ShiftNeck that learns the global feature relationship to generate a spatial bias that offsets the features. By concatenating the biased features with the unbiased, it can expand the channel width in a much cheaper way. This approach not only performs channel expansion efficiently but also enhances spatial perception cheaply by incorporating global information.

Based on the proposed concepts, we introduce **FreeNets**, which uses S2-Mixer and ShiftFFN for efficient token and channel mixing, respectively. Compared to existing networks which are full of separable operations, our FreeNets can achieve a superior speed-accuracy tradeoff, as depicted in Figure 1 (b). Furthermore, the utilization of S2-Mixer and Shift-FFN can also save the runtime memory requirements greatly by subtly exploiting the feature redundancy, as shown in Figure 1 (c). Within the compared methods, our model has two significant advantages: 1) we totally deprecate separable operations in the lightweight vision architecture, and 2) optimize the speed bottleneck of FFN, thereby having fewer memory access requirements and thus undoubtedly faster. Our contributions are as follows:

- We explore the potential of substituting separable operations with partially continuous operations, aiming to decrease execution time and save computational resources.
- We introduce a lightweight and fast token mixer called S2-Mixer. It is based on the proposed SWConv for cheaply depicting long-range pixel relationships through partial continuous operations.
- We introduce Shift-FFN as a swift, compact alternative to existing FFNs. It utilizes the global channel-wise relationship to shift features, cheaply broadening the channel width and aggregating global information.
- We introduce FreeNets, a family of vision backbones devoid of separable operations, offering better accuracy-efficiency tradeoffs against the latest efficient models.

Related Work

Efficient CNNs. The innovation of integrating Depthwise Separable Convolution (DWConv) (Chollet 2017) and Group Convolution (Ioannou et al. 2017) into advanced CNN architectures has led to the development of efficient CNNs, including MobileNets (Howard et al. 2017), ShuffleNets (Zhang et al. 2018; Ma et al. 2018), GhostNet (Han et al. 2020), and TVConv (Chen et al. 2022a). These advancements, by capitalizing on the concept of filter redundancy, have carved out a niche of efficient models highly suited for edge computing applications. Following these works, further research has delved into neural architecture search as a means to refine network structures, as exemplified by EfficientNets (Tan and Le 2019, 2021) and MobileNet-v3 (Howard et al. 2019). Additionally, the technique of reparameterization has also been explored to enhance the inference-time speed, a notable example being MobileOne (Vasu et al. 2023b). These methods predominantly focus on embedding strong vision inductive biases to efficiently decode local visual patterns with fewer parameters. However, this emphasis tends to limit their capability in identifying extensive pattern dependencies, often resulting in a performance gap when compared with the efficient

ViTs (Vasu et al. 2023a; Li et al. 2022b). Moreover, the extensive utilization of separable operations, although beneficial in minimizing the parameters, does not necessarily ensure higher efficiency on practical computing platforms.

Efficient ViTs. The ViT (Dosovitskiy et al. 2020) has quickly become a pivotal model. Subsequent research efforts have focused on refining ViT architecture. A significant trend is the development of efficient ViTs (Chen et al. 2022b; Li et al. 2022b; Mehta and Rastegari 2021; Vasu et al. 2023a; Shaker et al. 2023) that strike a favorable accuracy-speed trade-off. The primary strategy for creating efficient ViTs involves minimizing the computational complexity and training overhead of the SA mechanism. This is achieved through reducing the spatial dimensions within the SA process that capitalize on the spatial sparsity of images (Wang et al. 2022; Chu et al. 2021; Yu et al. 2022), integrating the efficient architecture from prior lightweight CNNs (Chen et al. 2022b; Li et al. 2022b; Mehta and Rastegari 2021), and adopting depth-wise convolutions for mixing with SA to reduce the complexity (Wu et al. 2021; Srinivas et al. 2021; Li et al. 2022a). These approaches seek to combine the global contextual awareness of ViTs with the efficiency and local processing strengths of CNNs. Nevertheless, as these ViTs increasingly integrate CNN components, their architecture begins to closely mirror the preceding efficient CNNs (Howard et al. 2017; Ma et al. 2018), yet still exhibit slower running speed. The inefficiency in practice lies in the SA mechanism, also a depth-wise operation that is hampered by frequent memory access issues due to its segmented computation across feature maps (or tokens).

Redundancy in Visual Features. The concept of feature redundancy widely exists in vision-related neural networks. Rational use of feature redundancy is one of the ways to improve model efficiency. In prior studies, the GhostNet (Han et al. 2020) first reveals the redundancy in convolution filters (or feature channels) and leverages the cheap operation to process features efficiently. Afterward, the recent FasterNets (Chen et al. 2023) also leverage the channel redundancy to accelerate the convolution inference speed. However, when compared with leading efficient models that employ strategies such as SA (Li et al. 2022b), reparameterization (Vasu et al. 2023a), and neural architecture search (Tan and Le 2021) to boost their capabilities, prior redundancy-based models (Han et al. 2020; Chen et al. 2023) significantly lag behind. In this paper, we explore the redundancy not merely at the channel level but also across spatial dimensions. By further considering the spatial sparsity in images, we further accelerate the inference and make our improved Conv operator excel at capturing long-range pixel dependencies like the powerful SA mechanism. Additionally, we delve into the redundancy present in channel mixers (FFNs) to further improve our accuracy-efficiency tradeoff.

Methodology

Sparse-wise Convolution

Here, we present the SWConv as a basic operator for efficient vision architectures. We start by presenting a progression from the vanilla Conv. Our goals are to (1) squeeze its parameters; (2) make it seem like SA captures long-range pixel dependencies for enhancing the representation capacity; and (3) maintain its continuous operation on channels to ensure processing speeds on practical devices.

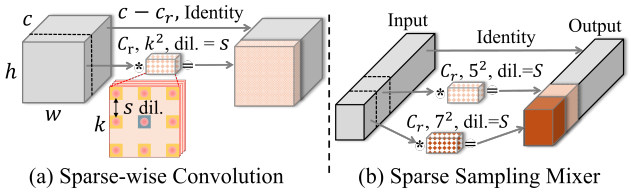

Figure 2: (a) Sparse-wise convolution; (b) Sparse sampling mixer (S2-Mixer). They are efficient by subtly sampling partially continuous signals to perform convolutional process.

Specifically, for an input feature $\mathbf{I} \in \mathbb{R}^{c\times h\times w}$, the vanilla bias-free convolution operation ($*$) can be defined as:

$$\mathbf{Y} = \mathbf{I} * \mathbf{f}^{\mathrm{c\times c\times k\times k}}, \quad \mathbf{Y} \in \mathbb{R}^{\mathrm{c\times h\times w}}, \tag{1}$$

where $\mathbf{f}$ is convolution filters with kernel size k; $\mathbf{Y}$ is the output feature. In Eq. (1), the receptive field of all filters is with size of k, and the local convolution process tends to build stronger dependencies between nearby continuous pixels (Raghu et al. 2021). These two factors imply that, compared to ViTs, CNNs are inefficient at learning global patterns with long pixel distances. As a result, using similar parameters, traditional CNNs are generally inferior to ViTs in tasks involving dense prediction and scene parsing, as highlighted in (Wang et al. 2022; Wu et al. 2022). Meanwhile, compared to the separable operations like DWConv, its parameters are much heavier for every learnable filter that is continuously performed across **all** the filters.

To address the above dilemma, we propose SWConv, a solution that is as lightweight as the DWConv while having significantly enhanced speed and superior capability in capturing global patterns. Specifically, we upgrade a primary Conv (Eq. (1)) to our SWConv in two steps:

(1). Leveraging the channel redundancy. For the input feature $\mathbf{I}$ with c channels, we reduce complexity by only applying our SWConv to the subset of continuous channels c_r with the ratio $r = c_r/c \in (0, 1)$. This step is effective as channel redundancy widely exists in image features, and the operation in partial channels has also been validated in prior works like GhostNet (Han et al. 2020) and FasterNet (Chen et al. 2023). By only seeing partial continuous channels, the parameter requirements can be effectively reduced, and operation speed can be further improved.

(2). Introducing the spatial sparsity. For each spatial map $\mathbf{S} \in \mathbb{R}^{h\times w}$ in the channel subset $\{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_{c_r}\}$ selected in step (1), SWConv only works on the pixels in the subset $\{h_s \times w_s\} \subset \{h \times w\}$, where the subset is selected from $\mathbf{S}$ by uniformly skipping pixels to make d dilations. By breaking the locality and retaining other visual inductive biases of Conv, this critical step empowers our SWConv to rapidly learn visual patterns similar to Conv while also excelling at capturing long-range pixel dependencies like MHSA. Also,

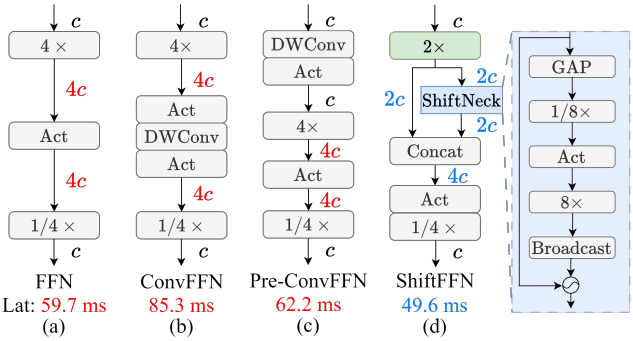

Figure 3: Block designs of existing FFNs and the proposed ShiftFFN. c indicates the channel dimension. The latency is measured using an input size of 384 $(c) \times 64 \times 64$ on the RTX-2080Ti GPU with restricted memory and I/O capacity.

we keep the selected pixels continuous across all the channels to ensure stable and fast on-device execution.

Based on the above upgrades, the SWConv can be illustrated in Figure 2 (a). It exhibits continuous movement across partial feature maps, thus having a faster running speed and lower operating memory requirement.

Spare Sampling Mixer

Building upon the SWConv, we further introduce the Spare Sampling Mixer (S2-Mixer) as a fast and lightweight token mixer solution. The detailed structure of the S2-Mixer is depicted in Figure 2 (b). Specifically, in the S2-Mixer, we apply multiple SWConvs with distinct kernel sizes to disjoint partial channels for capturing spatial patterns from different receptive fields. We define the default setting of S2-Mixer as employing two SWConvs ($d = 2$, $r = 1/8$) with template kernel sizes of 5×5 and 7×7, enabling spatial sparsity and diversified receptive fields. This setup mimics the SA mechanism's ability to detect long-range dependencies while maintaining the continuous operation and inductive biases of traditional convolutions, suggesting the S2-Mixer as an alternative for token mixing in visual applications.

Moreover, the S2-Mixer processes only a partial of the signals within the features, leaving a segment of signals from the previous layer unchanged. This is equivalent to the residual feature effect. Thus, we streamline the existing token-channel mixer architecture (Yu et al. 2022) into a single shortcut branch, by eliminating the residual branch between the S2-Mixer and the subsequent channel mixer (see Figure 4). This adjustment could offer further advantages in speed and memory usage over traditional two-shortcut designs.

Shift Feed-Forward Network

The ShiftFFN serves as a faster substitution to FFN for channel mixing. It exhibits faster running speed while also excelling in aggregating global information. Given the input feature $\mathbf{I}_{in} \in \mathbb{R}^{c \times h \times w}$, we first define the original FFN as:

$$\mathbf{I}_{out} = \mathrm{GELU}(\mathbf{I}_{\mathrm{in}} * \mathbf{W}_{\mathrm{in}}^{4c \times c \times 1 \times 1}) * \mathbf{W}_{\mathrm{out}}^{c \times 4c \times 1 \times 1}, \quad (2)$$

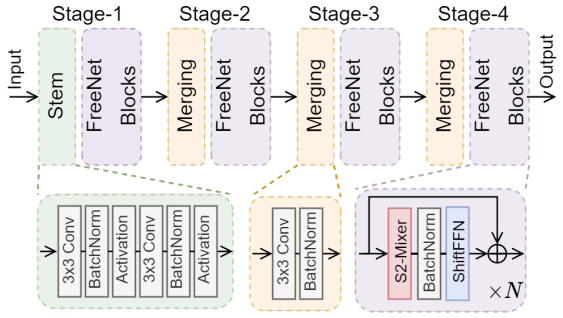

Figure 4: Overview of the FreeNet architecture.

Variant	#Channels	#Blocks	#FFN ratios	#Param
FreeNet-S1	[32, 64, 144, 256]	[1, 2, 4, 2]	[2, 2, 2, 2]	$2.4M$
FreeNet-S2	[48, 96, 224, 384]	[1, 2, 8, 2]	[2, 2, 2, 2]	$6.8M$
FreeNet-S3	[48, 96, 224, 384]	[2, 3, 12, 3]	[2, 2, 2, 2]	$9.6M$
FreeNet-S4	[64, 128, 272, 512]	[2, 3, 12, 3]	[2, 2, 2, 2]	$15.3M$

Table 1: Configuration of four FreeNet variants. Channels: number of channels per stage; Blocks: number of FreeNet blocks per stage. FFN ratio: channel expand ratio n of the first projection (1×1 Conv) layer in ShiftFFNs per stage.

where $\mathbf{W}_{in}$ and $\mathbf{W}_{out}$ are two linear projection layers used to first increase the channel width from c to $4c$ and then decrease it back to c. The block design of FFN is shown in Figure 3 (a). Using such an inverted residual scheme to increase the peak channel width is necessary for improving the accuracy, but it will impede the running speed due to the substantial memory access required (Chen et al. 2023). Furthermore, in the vanilla FFN, there is no spatial-wise operation. Thus, to enhance the capacity for aggregation of global spatial information, many works (Wu et al. 2022; Wang et al. 2022; Vasu et al. 2023a; Shaker et al. 2023) propose to use additional DWConvs within (ConvFFN) or before (Pre-ConvFFN) the FFN, as shown in Figure 3 (b) and (c). While these designs perform better than the traditional FFN, they bring varying degrees of additional computational overhead.

Thus, towards a cost-effective design, we propose a ShiftFFN that has a faster speed and lower memory consumption than the existing FFN designs. It also excels in global information aggregation. The block design is shown in Figure 3 (d). Our thinking is concise but effective. We first reduce the computation and memory access by cutting the projection ratio of the first PWConv from $4\times$ to $2\times$. Then, a shiftNeck is designed to learn global information efficiently like the SENet (Hu, Shen, and Sun 2018) on the copy of projected features. To remedy the loss in the channel width, we simply use the learned global information to bias the original features and concatenate it with the unbiased one to increase the channel width to $4\times$ cheaply. In this manner, we switch the original channel projection $(c \rightarrow 4c \rightarrow c)$ to the proposed scheme $(c \rightarrow 2c \rightarrow 4c \rightarrow c)$ for acceleration, where the process $2c \rightarrow 4c$ is fast as we exploit the copy-shift-concat operation flow to increase the channel width. Meanwhile, as the ShiftNeck can cheaply aggregate the global in-

Modules ×5	Input size (c×h×w)	FLOPs (M)	Latency (ms)↓	Thp (fps)↑	Operators ×5	Input size (c×h×w)	FLOPs (M)	Latency (ms)↓	Thp (fps)↑
FFN	48x512x512	4832	297.7	51.4	7×7 Conv	48x512x512	154168	386.5	41.4
	96×256×256	4832	148.8	102.2		96×256×256	155010	266.5	60.0
	192×128×128	4832	89.2	179.4		192×128×128	162245	214.4	74.6
	384×64×64	4832	59.7	271.6		384×64×64	177309	214.8	74.5
	512×32×32	2147	25.6	688.4		512×32×32	93255	114.8	139.4
	1024×16×16	2147	19.7	903.2		1024×16×16	126395	166.8	95.9
	2048×8×8	2147	17.3	923.5		2048×8×8	209631	282.9	56.6
	Average	3681	94	446		Average	154002	235	77
ConvFFN (Wu et al. 2022)	48×512×512	26424	478.4	33.4	7×7 DWConv (Liu et al. 2022)	48×512×512	3155	248.1	64.5
	96×256×256	25292	242.9	65.9		96×256×256	1614	126.6	126.4
	192×128×128	24725	135.8	117.9		192×128×128	845	66.6	240.2
	384×64×64	24442	85.3	187.6		384×64×64	461	36.1	443.0
	512×32×32	10832	31.9	501.9		512×32×32	182	14.3	1118.9
	1024×16×16	10785	21.9	730.5		1024×16×16	123	9.4	1697.6
	2048×8×8	10761	19.3	830.2		2048×8×8	102	7.8	2059.7
	Average	19037	145	352		Average	926	73	821
Pre-ConvFFN (Shaker et al. 2023)	48×512×512	24725	315.1	51.0	Linear MHSA (Wang et al. 2022)	48×512×512	12270	467.5	35.4
	96×256×256	24442	158.7	99.7		96×256×256	9160	145.4	112.3
	192×128×128	24301	95.9	169.4		192×128×128	7624	56.3	287.5
	384×64×64	24230	62.2	256.5		384×64×64	6927	25.5	625.2
	512×32×32	10761	23.5	676.4		512×32×32	3137	8.3	1912.8
	1024×16×16	10749	17.8	896.6		1024×16×16	3586	6.5	2438.0
	2048×8×8	10743	18.5	853.6		2048×8×8	5834	9.8	1635.7
	Average	18565	99	429		Average	6934	103	1007
ShiftFFN (Ours)	48×512×512	18245	272.3	55.0	S2-Mixer (Ours)	48×512×512	1604	38.0	420.5
	96×256×256	18182	125.5	115.0		96×256×256	1604	13.3	1199.0
	192×128×128	18151	69.3	209.6		192×128×128	1604	6.2	2576.3
	384×64×64	18135	49.6	286.9		384×64×64	1604	4.1	3458.2
	512×32×32	8059	16.5	845.7		512×32×32	713	1.3	11756.2
	1024×16×16	8060	15.0	1000.6		1024×16×16	713	1.2	13962.1
	2048×8×8	8075	12.6	1344.7		2048×8×8	713	1.2	13963.7
	Average	13844	**80**	**551**		Average	1222	**9**	**6762**

Table 2: Quantitative speed comparisons with existing token mixer (left) and channel mixer (right) solutions. All the latency and throughput metrics (Thp.) are measured on one RTX-2080Ti without further acceleration.

formation, the ShiftFFN enables our model to perform efficiently in dense prediction and scene parsing tasks.

Overall Architectures

Leveraging the S2-Mixer and ShiftFFN, we present FreeNets, a family of efficient backbones with a highly streamlined architecture, as depicted in Figure 4. FreeNets feature four hierarchical stages, each comprising a stack of FreeNet blocks, where each block integrates an S2-Mixer followed by a ShiftFFN. A single shortcut is implemented per block since the S2-Mixer retains partial information from previous layers. To address varying computational budgets, we provide FreeNets in four variants—S1, S2, S3, and S4—detailed in Table 1. For comprehensive configurations and further discussions, please refer to the appendix.

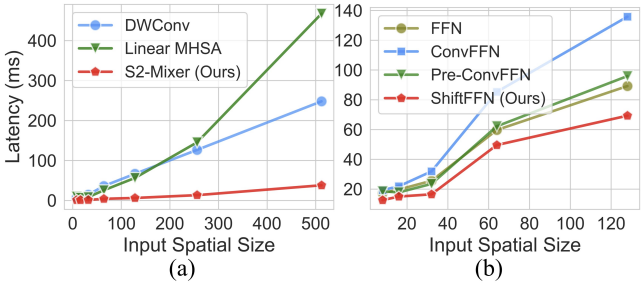

Figure 5: Visualization of latency vs. input spatial size on efficient token mixers in (a) and existing channel mixer in (b). The proposed S2-Mixer and ShiftFFN run consistently faster than existing solutions on various input spatial sizes.

Experiments

During experiments, we use NVIDIA GPU throughput and latency (as presented in Tables 2 - 5), along with ARM mobile GPU and CPU latency (in Table 3) as on-device efficiency metrics. For the speed benchmark on NVIDIA GPU, we select the RTX-2080ti GPU due to its weak computational power and CUDA memory I/O capacity, which provides a clearer understanding of computational bottlenecks, particularly for scenarios requiring lightweight models. The on-mobile speed benchmark is conducted on the Snapdragon 8cx Gen 3 chip using the Tencent TNN Android inference framework (CPU part) and OpenGL library (GPU part).

Quantitative Analysis on Speed

We first show the computational efficiency by comparing the S2-Mixer and ShiftFFN with their competitors in FLOPs, throughput, and latency. All the metrics are computed by stacking the examined operator or block 5 times and taking feature maps with various dimensions as the input.

The quantitative analysis is listed in Table 2. We also

Model	Res. (px.)	Param (M)	FLOPs (G)	GPU Throughput (fps)↑	GPU Latency (ms)↓	M-GPU Latency (ms)↓	M-CPU Latency (ms)↓	Top-1 Acc (%)↑
MobileViT-XXS (Mehta and Rastegari 2021)	256	1.3	0.42	1724	18.55	–	24.1	69.0
EdgeNeXt-XXS (Maaz et al. 2022)	256	1.3	0.26	1876	17.05	–	6.9	71.2
MobileOne-S0† (Vasu et al. 2023b)	224	2.1	0.27	4433	7.21	3.3	**4.4**	71.4
FasterNet-T0 (Chen et al. 2023)	224	3.9	0.34	4707	4.88	6.3	9.2	71.9
MobileNetV2 (Sandler et al. 2018)	224	3.5	0.30	2622	12.23	6.5	8.9	72.0
StarNet-S1 (Ma et al. 2024)	224	2.9	0.42	2800	11.42	3.8	5.1	73.5
StarNet-S2 (Ma et al. 2024)	224	3.7	0.55	2583	12.38	5.7	8.9	74.8
SwiftFormer-XS (Shaker et al. 2023)	224	3.5	0.60	2122	15.07	–	11.9	75.7
MobileOne-S1† (Vasu et al. 2023b)	224	4.8	0.82	2342	13.66	5.9	11.6	**75.9**
FreeNet-S1	224	2.4	0.28	**4931**	**6.22**	**3.1**	**4.4**	70.7/71.9
EdgeNeXt-XS (Maaz et al. 2022)	256	2.3	0.54	1451	22.04	–	20.5	75.0
FasterNet-T1 (Chen et al. 2023)	224	7.6	0.85	2151	16.71	7.3	16.4	76.2
MobileOne-S2† (Vasu et al. 2023b)	224	7.8	1.29	1761	18.16	8.4	18.9	77.4
StarNet-S4 (Ma et al. 2024)	224	7.5	1.07	1294	24.72	8.4	18.1	78.4
SwiftFormer-S (Shaker et al. 2023)	224	6.1	0.93	1687	19.29	–	21.6	78.5
EfficientNet-B1∗ (Tan and Le 2019)	256	7.8	0.70	994	32.16	15.3	34.6	**79.1**
FreeNet-S2	224	6.8	0.90	**2165**	**16.69**	**7.1**	**15.4**	77.0/78.8
MobileOne-S3† (Vasu et al. 2023b)	224	10.1	1.89	1360	23.52	9.1	26.7	78.1
MobileViT-S (Mehta and Rastegari 2021)	256	5.6	2.03	776	41.22	–	62.7	78.4
EdgeNeXt-S (Maaz et al. 2022)	256	5.6	1.26	1032	31.01	–	30.1	79.4
FastViT-S12† (Vasu et al. 2023a)	256	8.8	1.80	1261	25.36	16.8	59.3	**79.8**
FreeNet-S3	224	9.6	1.31	**1593**	**23.29**	**8.8**	**25.3**	78.6/79.4
FasterNet-T2 (Chen et al. 2023)	224	15.0	1.90	**1500**	**20.95**	13.2	37.1	78.9
MobileOne-S4† (Vasu et al. 2023b)	224	14.8	2.97	890	35.94	15.6	43.4	79.4
EfficientNet-B2∗ (Tan and Le 2019)	288	9.2	1.00	752	42.53	16.9	58.9	80.1
FastViT-SA12† (Vasu et al. 2023a)	256	10.9	1.90	987	28.99	–	64.6	80.6
SwiftFormer-L1 (Shaker et al. 2023)	224	12.1	1.60	1211	26.41	–	43.9	80.9
FreeNet-S4	224	15.3	2.01	1354	22.48	**12.6**	**36.4**	79.4/**81.0**

Table 3: Comparison with efficient models on the ImageNet-1K benchmark. We report the Top-1 accuracy under both basic and advanced (left and right side of "/") training techniques. "$*$" and † marks indicate models using architecture search and reparameterization for efficiency-boosting, respectively. M-GPU and M-CPU means On-Mobile GPU and CPU speed benchmarks, respectively. The "$-$" means implementation not support by the OpenGL library, thus no mobile GPU benchmark results.

graphically represent latency in comparison to input size (h$\times$w) in Figure 5. Evidently, our ShiftFFN and S2-Mixer demonstrate superior on-device efficiency over current operators. Specifically, the average throughput of our ShiftFFN is 1.28$\times$, 1.56$\times$, and 1.23$\times$ faster than Pre-ConvFFN (Shaker et al. 2023), ConvFFN (Wang et al. 2022), and basic FFN, respectively. The S2-Mixer significantly outperforms existing learnable token mixers, achieving an average throughput that is 11$\times$, 8$\times$, and 87$\times$ higher than Linear MHSA (Wang et al. 2022), 7$\times$7 DWConv (Liu et al. 2022) and 7$\times$7 Conv, respectively. This heightened efficiency is noteworthy, especially since our S2-Mixer is built using two SWConvs with large kernel sizes. These results confirm that, compared to existing operators, the enhancements in latency and throughput achieved by the proposed S2-Mixer and Shift-FFN are consistent across various input shapes.

Evaluation on ImageNet-1K Classification

We benchmark FreeNets on the ImageNet-1k dataset (Deng et al. 2009). To fairly compare with existing models using different training techniques, our FreeNets were trained following two settings commonly used in prior works: 1) basic settings following (Chen et al. 2023) and 2) advanced settings with distillation (Touvron et al. 2021; Shaker et al. 2023). We report the performance under both settings in Table 3. Briefly, all models are trained for 300 epochs using the AdamW optimizer, with a learning rate scaled as $\frac{\text{Batch Size}}{1024}$ $\times$ 1e-3. Our models are implemented in PyTorch and trained using 8 AMD Instinct MI250X GPUs with the MXNet-REC format data. For detailed settings, results using more training techniques (Leclerc et al. 2023) and higher resolutions (e.g., $256px$), please refer to our appendix.

Results. Table 3 shows the efficiency of FreeNets compared to prior methods. Compared to FasterNet (Chen et al. 2023), which uses feature redundancy for efficient token mixing, FreeNets deliver better accuracy-speed tradeoffs by further incorporating spatial sparsity and addressing FFN speed bottlenecks. For example, under the comparable basic settings, FreeNet-S2 (77.0%) and FreeNet-S4 (79.4%) surpass FasterNet-T1 (76.2%) and FasterNet-T2 (78.9%) by 0.8% and 0.5%, respectively, while offering faster speed on mobile GPU and CPU devices. Using advanced training techniques to further boost performance, FreeNet-S2 (78.8%) outperforms StarNet-S4 (78.4%) by 0.4% in accuracy while delivering 40% higher GPU throughput. Similar advantages are evident over MobileOne (Vasu et al. 2023b). Moreover, FreeNet-S4 exceeds SwiftFormer-L1 (Shaker et al. 2023) in accuracy and achieves over 10% faster GPU throughput. These findings confirm the effectiveness of the proposed token/channel mixing strategies compared to traditional DWConv-prior approaches, especially in resource-constrained environments like mobile devices.

Dense Prediction and Scene Parsing

We evaluate the ImageNet-1k pretrained FreeNet on the MS-COCO dataset (Lin et al. 2014) and ADE20K datasets (Zhou

Backbone	Param	Latency	Detection and Instance Segmentation						Semantic
			AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}	mIoU(%)
ResNet-50 (He et al. 2016)	25.5	119.0	38.0	58.6	41.4	34.4	55.1	36.7	36.7
PoolFormer-S12 (Yu et al. 2022)	11.9	113.7	37.3	59.0	40.1	34.6	55.8	36.9	37.2
FastViT-SA12 (Vasu et al. 2023a)	10.9	133.8	38.9	60.5	42.2	35.9	57.6	38.1	38.0
MobileOne-S3 (Vasu et al. 2023b)	10.4	99.4	-	-	-	-	-	-	36.2
FreeNet-S4	15.3	119.7	**39.1**	**60.8**	**42.5**	**36.5**	**58.9**	**38.8**	**38.5**

Table 4: Results on dense prediction/scene parsing tasks. We use the Mask-RCNN with a 1× training schedule for object detection/instance segmentation on the MS-COCO dataset (Lin et al. 2014). The semantic segmentation is performed on the ADE20K dataset (Zhou et al. 2017) using the Semantic FPN decoder. GPU latency is measured using the input size of 512px.

et al. 2017) for object detection/instance segmentation and semantic segmentation, respectively. For object detection and instance segmentation, we employ the Mask-RCNN (He et al. 2017) framework following (Vasu et al. 2023a). For semantic segmentation, we leveraged the semantic FPN decoder and trained it according to widely accepted settings (Vasu et al. 2023a; Yu et al. 2022). Please see detailed settings and more results in the appendix.

Results. As illustrated in Table 4, the proposed FreeNet demonstrates a superior accuracy-speed tradeoff in dense prediction and scene parsing tasks. The FreeNet-S4 exceeds the FastViT-SA12 (Vasu et al. 2023a) by improvements of 0.2%, 0.3%, and 0.3% in AP^b, AP^b_{50}, and AP^b_{75} scores, respectively, along with a reduction in latency by over 10%. These results underscore the efficiency and precision of our S2-Mixer and ShiftFFN in tackling the challenging dense prediction and scene parsing tasks. This is because the S2-Mixer can cheaply build long-range pixel dependency to expand the receptive field, and the ShiftFFN aggregates global information during channel mixing. Both contribute significantly to the dense pixel/scene understanding.

Ablation Analysis

We conduct ablation studies that replace existing token/channel mixers with our proposed. The ConvNeXt and PVTv2 are chosen as the objective models. ConvNeXt (Liu et al. 2022) uses the 7×7 DWConv as the token mixer and follows the standard FFN for channel mixing; PVTv2 (Wang et al. 2022) employs the linear SA for token mixing and uses the ConvFFN for channel mixing. In Table 5, we replace their token and channel mixers separately to show the variations in parameters, FLOPs, GPU latency, and ImageNet-1K accuracy. We aim to verify the efficiency and effectiveness of the proposed components on existing architectures. Therefore, we selectively replace only partial operators, specifically those in the final stage of the network.

Results are presented in Table 5. The efficiency of ConvNeXt-T and PVTv2-b2-li improves significantly when replacing their default operators with our proposed solutions. Specifically, replacing the FFN in the final stage of ConvNeXt-T with our ShiftFFN increases running speed with negligible impact on accuracy. Similarly, replacing the 7×7 DWConv boosts all speed metrics while maintaining accuracy. Comparable improvements are observed in PVTv2 when our efficient solutions replace the linear self-attention (Li-SA) and ConvFFN individually. Additionally, we conduct "parameter alignment" experiments by replacing both

Basline & Modification	Param (M)	FLOPs (G)	Latency (ms)↓	IN-1K Top-1(%)↑
ConvNeXt-T (Liu et al. 2022)	28.6	4466	44.0	82.119
FFNs in the last stages → ShiftFFNs	26.2	4292	43.1	82.123
Performance variations	**↓2.4**	**↓174**	**↓0.9**	**↑0.004**
ConvNeXt-T (Liu et al. 2022)	28.6	4466	44.0	82.119
DWConvs in the last stage → S2-Mixers	28.6	4474	42.1	82.131
Performance variations	–	**↑8**	**↓1.9**	**↑0.012**
PVTv2-b2-li (Wang et al. 2022)	22.6	3909	54.1	82.092
ConvFFNs the last stages → ShiftFFNs	21.3	3827	49.2	82.089
Performance variations	**↓1.3**	**↓82**	**↓4.9**	**↓0.003**
PVTv2-b2-li (Wang et al. 2022)	22.6	3909	54.1	82.092
Li-SAs in the last stage → S2-Mixers	19.5	3743	48.0	82.126
Performance variations	**↓3.1**	**↓166**	**↓6.2**	**↑0.034**
PVTv2-b2-li (Wang et al. 2022)	22.6	3909	54.1	82.092
Last Stage: ShiftFFN & S2-Mixer	18.2	3651	43.2	82.104
Depth: [3,4,6,3] → [3,3,8,4]	22.5	3934	47.9	83.259
Performance variations	**↓0.1**	**↑25**	**↓6.1**	**↑ 1.167**

Table 5: Ablation studies on using S2-Mixer and ShiftFFN to replace the token and channel mixers in ConvNeXt-T (Liu et al. 2022) and PVTv2-b2-li (Wang et al. 2022).

the channel and token mixers in the last stage of PVTv2-b2-li and increasing the depth to match the original parameters. As shown in the last row of Table 5, with similar parameters and FLOPs, our solutions accelerate the baseline by 6.1 ms while improving accuracy by around 1%. These results validate the effectiveness of the S2-Mixer and ShiftFFN, particularly in the final network stages, where feature redundancy is relatively prevalent.

Conclusion

This paper revisits the design of efficient models, identifying the excessive dependence on discontinuous depth-wise separable operations as a bottleneck to achieving higher efficiency. In pursuit of an enhanced accuracy-speed tradeoff, we propose the S2-Mixer as a continuous alternative for mixing tokens faster. The S2-Mixer, conceptualized through a lens of sparse sampling, not only enhances processing speed but also excels in capturing long-range pixel dependencies. Additionally, we introduce the ShiftFFN, an enhancement over existing FFN solutions, which improves global information aggregation and delivers faster inference speeds. Leveraging the proposed components, we present FreeNets: a new suite of models devoid of separable operations, setting new benchmarks for speed and accuracy across a spectrum of vision tasks with less memory consumption.

Acknowledgements

This work was supported by the Research Council of Finland (former Academy of Finland) Academy Professor project EmotionAI (grants 336116, 345122, 359854), ICT 2023 project TrustFace (grant 345948), the University of Oulu & Research Council of Finland Profi 7 (grant 352788), and Academy of Finland Flagship program: the Finnish Center for Artificial Intelligence FCAI. As well, the authors wish to acknowledge CSC – IT Center for Science, Finland, for computational resources.

References

Chen, J.; He, T.; Zhuo, W.; Ma, L.; Ha, S.; and Chan, S.-H. G. 2022a. Tvconv: Efficient translation variant convolution for layout-aware visual processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12548–12558.

Chen, J.; Kao, S.-h.; He, H.; Zhuo, W.; Wen, S.; Lee, C.-H.; and Chan, S.-H. G. 2023. Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks. In *CVPR*, 12021–12031.

Chen, Y.; Dai, X.; Chen, D.; Liu, M.; Dong, X.; Yuan, L.; and Liu, Z. 2022b. Mobile-former: Bridging mobilenet and transformer. In *CVPR*, 5270–5279.

Chollet, F. 2017. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 1251–1258.

Chu, X.; Tian, Z.; Wang, Y.; Zhang, B.; Ren, H.; Wei, X.; Xia, H.; and Shen, C. 2021. Twins: Revisiting the design of spatial attention in vision transformers. In *NeurIPS*, volume 34, 9355–9366.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Han, K.; Wang, Y.; Chen, H.; Chen, X.; Guo, J.; Liu, Z.; Tang, Y.; Xiao, A.; Xu, C.; Xu, Y.; et al. 2022. A survey on vision transformer. *IEEE TPAMI*, 45(1): 87–110.

Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; and Xu, C. 2020. Ghostnet: More features from cheap operations. In *CVPR*, 1580–1589.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *ICCV*, 2961–2969.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. 2019. Searching for mobilenetv3. In *ICCV*, 1314–1324.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7132–7141.

Ioannou, Y.; Robertson, D.; Cipolla, R.; and Criminisi, A. 2017. Deep roots: Improving cnn efficiency with hierarchical filter groups. In *CVPR*, 1231–1240.

Leclerc, G.; Ilyas, A.; Engstrom, L.; Park, S. M.; Salman, H.; and Madry, A. 2023. FFCV: Accelerating training by removing data bottlenecks. In *CVPR*, 12011–12020.

Li, K.; Wang, Y.; Gao, P.; Song, G.; Liu, Y.; Li, H.; and Qiao, Y. 2022a. Uniformer: Unified transformer for efficient spatiotemporal representation learning. *arXiv preprint arXiv:2201.04676*.

Li, Y.; Yuan, G.; Wen, Y.; Hu, J.; Evangelidis, G.; Tulyakov, S.; Wang, Y.; and Ren, J. 2022b. Efficientformer: Vision transformers at mobilenet speed. In *NeurIPS*, volume 35, 12934–12949.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740–755. Springer.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, 10012–10022.

Liu, Z.; Mao, H.; Wu, C.-Y.; Feichtenhofer, C.; Darrell, T.; and Xie, S. 2022. A convnet for the 2020s. In *CVPR*, 11976–11986.

Lu, G.; Zhang, W.; and Wang, Z. 2021. Optimizing depthwise separable convolution operations on gpus. *IEEE Transactions on Parallel and Distributed Systems*, 33(1): 70–87.

Ma, N.; Zhang, X.; Zheng, H.-T.; and Sun, J. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 116–131.

Ma, X.; Dai, X.; Bai, Y.; Wang, Y.; and Fu, Y. 2024. Rewrite the Stars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5694–5703.

Maaz, M.; Shaker, A.; Cholakkal, H.; Khan, S.; Zamir, S. W.; Anwer, R. M.; and Shahbaz Khan, F. 2022. Edgenext: efficiently amalgamated cnn-transformer architecture for mobile vision applications. In *ECCV*, 3–20. Springer.

Mehta, S.; and Rastegari, M. 2021. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*.

Raghu, M.; Unterthiner, T.; Kornblith, S.; Zhang, C.; and Dosovitskiy, A. 2021. Do vision transformers see like convolutional neural networks? In *NeurIPS*, volume 34, 12116–12128.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 4510–4520.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 618–626.

Shaker, A.; Maaz, M.; Rasheed, H.; Khan, S.; Yang, M.-H.; and Khan, F. S. 2023. SwiftFormer: Efficient Additive Attention for Transformer-based Real-time Mobile Vision Applications. *arXiv preprint arXiv:2303.15446*.

Srinivas, A.; Lin, T.-Y.; Parmar, N.; Shlens, J.; Abbeel, P.; and Vaswani, A. 2021. Bottleneck transformers for visual recognition. In *CVPR*, 16519–16529.

Tan, M.; and Le, Q. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, 6105–6114. PMLR.

Tan, M.; and Le, Q. 2021. Efficientnetv2: Smaller models and faster training. In *ICML*, 10096–10106. PMLR.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2021. Training data-efficient image transformers & distillation through attention. In *ICML*, 10347–10357. PMLR.

Vasu, P. K. A.; Gabriel, J.; Zhu, J.; Tuzel, O.; and Ranjan, A. 2023a. FastViT: A Fast Hybrid Vision Transformer using Structural Reparameterization. *arXiv preprint arXiv:2303.14189*.

Vasu, P. K. A.; Gabriel, J.; Zhu, J.; Tuzel, O.; and Ranjan, A. 2023b. MobileOne: An Improved One Millisecond Mobile Backbone. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7907–7917.

Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; and Shao, L. 2021. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, 568–578.

Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; and Shao, L. 2022. Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3): 415–424.

Wu, H.; Xiao, B.; Codella, N.; Liu, M.; Dai, X.; Yuan, L.; and Zhang, L. 2021. Cvt: Introducing convolutions to vision transformers. In *ICCV*, 22–31.

Wu, Y.-H.; Liu, Y.; Zhan, X.; and Cheng, M.-M. 2022. P2T: Pyramid pooling transformer for scene understanding. *IEEE TPAMI*.

Yu, W.; Luo, M.; Zhou, P.; Si, C.; Zhou, Y.; Wang, X.; Feng, J.; and Yan, S. 2022. Metaformer is actually what you need for vision. In *CVPR*, 10819–10829.

Yun, S.; and Ro, Y. 2024. Shvit: Single-head vision transformer with memory efficient macro design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5756–5767.

Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 6848–6856.

Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; and Torralba, A. 2017. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 633–641.