LowFormer: Hardware Efficient Design for Convolutional Transformer Backbones

LowFormer: Hardware Efficient Design for Convolutional Transformer Backbones

Moritz Nottebaum[1]
nottebaum.moritz@spes.uniud.it

Matteo Dunnhofer[1]
matteo.dunnhofer@uniud.it

Christian Micheloni[1]
christian.micheloni@uniud.it

[1]University of Udine, Italy

Abstract

Research in efficient vision backbones is evolving into models that are a mixture of convolutions and transformer blocks. A smart combination of both, architecture-wise and component-wise is mandatory to excel in the speed-accuracy trade-off. Most publications focus on maximizing accuracy and utilize MACs (multiply accumulate operations) as an efficiency metric. The latter however often do not measure accurately how fast a model actually is due to factors like memory access cost and degree of parallelism. We analyzed common modules and architectural design choices for backbones not in terms of MACs, but rather in actual throughput and latency, as the combination of the latter two is a better representation of the efficiency of models in real applications. We applied the conclusions taken from that analysis to create a recipe for increasing hardware-efficiency in macro design. Additionally we introduce a simple slimmed-down version of Multi-Head Self-Attention, that aligns with our analysis. We combine both macro and micro design to create a new family of hardware-efficient backbone networks called LowFormer. LowFormer achieves a remarkable speedup in terms of throughput and latency, while achieving similar or better accuracy than current state-of-the-art efficient backbones. In order to prove the generalizability of our hardware-efficient design, we evaluate our method on GPU, mobile GPU and ARM CPU. We further show that the downstream tasks object detection and semantic segmentation profit from our hardware-efficient architecture. Code and models are available at https://github.com/altair199797/LowFormer.

1. Introduction

Recent research in efficient vision backbone networks has focused on combining attention mechanisms with convolutional layers. The mixture of local information extraction (by convolutions) and global reasoning (by attention) has proven to be superior to homogeneous models [2, 24, 30, 47]. Besting the speed-accuracy trade-off has been the primary goal of research in that domain, in order to subsequently improve the efficiency of downstream tasks, like object detection, pose estimation and semantic segmentation [13, 19]. Efficiency in backbone architectures is also particularly important to improve the applicability of downstream tasks on mobile and edge devices [39].

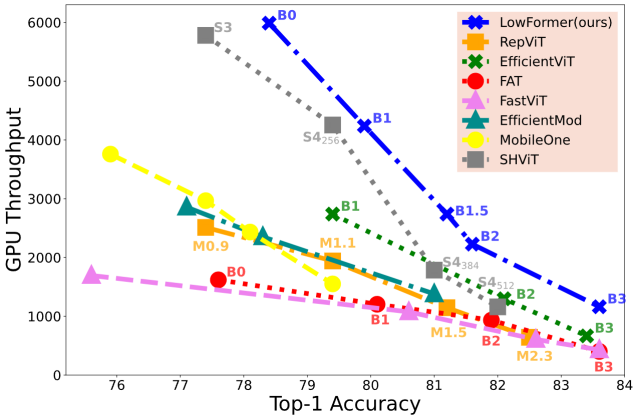

Figure 1. Comparison of GPU throughput and top-1 accuracy of recent image classification architectures and LowFormer. The markers refer to different complexity classes of corresponding architecture families. LowFormer consistently achieves a higher throughput than models with similar accuracy.

To measure the computational efficiency of deep learning models and compare them, it is common to count the amount of MACs (multiply and accumulate operations) [34]. The number of MACs of a model also strongly correlates with its accuracy. For example, when increasing them by width, depth or resolution scaling, the accuracy can be improved [34]. However MACs are not created equal and ignore factors like memory access cost and the degree of

parallelism [9, 24, 27, 39], which can have a substantial effect on execution time of a model.

Our goal is to improve throughput and latency for efficient backbone architectures while increasing accuracy by investigating when MACs are executed most efficiently. We subsequently deduct recipes for efficient architectural design, that follow these findings. For our investigation we carry out several experiments in Section 3 that contrast execution time and number of MACs of macro design decisions.

We apply the findings to create a new family of backbone networks called LowFormer. The exploitation of efficient MAC execution due to our macro design makes LowFormer faster and more accurate than previously proposed models (see Figure 1). Following the results of our investigation we complement our architecture with a simple and lightweight adaption of the traditional attention [40], that downsamples and upsamples the feature maps around the Scaled Dot-Product Attention (SDA). As a result the SDA is operating on a lower input resolution, hence the name **Low**Former.

We confirm the generalizability of the hardware efficiency of LowFormer by measuring GPU throughput, GPU latency, mobile GPU latency and ARM CPU latency. We back up our design decisions by an extensive ablation study (see Section 5.3). Our proposed architecture has a simple micro and macro design and allows us to scale it from low complexity(LowFormer-B0) to higher complexity(LowFormer-B3). In total we feature five models (B0,B1,B1.5,B2,B3) and our top-1 accuracy on ImageNet-1K [10] ranges from 78.4% to 83.64%. For example compared to MobileOne-S2 [39], LowFormer-B0 has 2× the throughput and 15 % less latency on GPU, while scoring 1% better in top-1 accuracy. Our model LowFormer-B3, featuring the highest complexity, almost has 3× the GPU throughput and 55% of the GPU latency of FAT-B3 [12].

We also integrate LowFormer into Semantic FPN [18] and RetinaNet [23] to improve the efficacy of semantic segmentation and object detection models. Within the first framework for example, our LowFormer-B1 backbone achieves 1.7% better mIoU than FastViT-SA12 [38], while having 3× the throughput and 30 % less latency on GPU. In summary, our contributions are as follows:

- We present a new backbone architecture family with a hardware-efficient macro design and a new lightweight attention. Our models are faster in terms of throughput and latency compared to models with similar accuracy.
- On several computing devices we carry out an exhaustive speed analysis of convolutions in different configurations to contrast amount of MAC operations and measured execution time.
- We show that LowFormer generalizes well to semantic segmentation and object detection, as well as retains its speed-up on different hardware like Mobile GPU and ARM CPU.

2. Related Work

Hardware-Efficient Model Design. Achieving the highest accuracy at all (computational) cost has long since ceased to be the only goal in deep learning [34]. An ever-increasing share of research focuses to create the most efficient architecture and subsequently to achieve the best speed-accuracy trade-off [6, 17, 31]. Earlier approaches mainly equated speed with a minimal amount of MACs [5, 15, 36], while more recent research increasingly judges models by throughput or latency on for example desktop GPU and CPU [2, 28, 47], mobile GPU [39] or generally edge devices [6].

Memory access cost and degree of parallelism have become important factors for efficient model design. Methods like EfficientViT [24] remark that the Multi-Head Self-Attention (MHSA) induces higher memory access cost than the Feed-Forward Network (FFN) in the transformer block. Therefore they increase the ratio of FFN compared to MHSA in their architecture, with minimal accuracy loss. The authors of MobileOne [39] on the other side analyze the effect of activation functions and multi-branch architectures on mobile latency. ShuffleNetV2 [28] and FasterNet [3] both point out that grouped convolutions are inefficiently executed on current hardware due to high memory access cost [3]. We follow their insights, but instead of ungrouping a portion of the convolutions [28] or introducing a new micro design [3], we study how fusing depthwise and pointwise convolutions affects execution time. We further analyze the effect of resolution on hardware-efficiency and apply the insights of both studies in our macro design.

Efficient Attention. There is a jungle of different kind of attention-like operations, that try to replace the traditional attention operation. Many remove its quadratic nature by variations of linear attention [1, 2, 37] based on Wang et al.[42]. However Yu et al.[46] showed that attention in itself is not as important as we thought and can even be replaced by a simple pooling operation. Li et al.[20] took that idea further and used the efficient pooling operation for the first 3 stages and the traditional attention for the last 2 stages [20]. Others [11, 43, 45] downsample the keys K and V before the attention operation, either with convolutions or pooling . Si et al.[33] on the other side also downsamples Q, therefore completely operating on a lower resolution. This is similar to how our attention approach works, however we use convolutions for downsampling instead of pooling and do not incorporate their inception token mixer, which separates the channels for multiple paths, one of them being the attention on a lower resolution. Additionally in contrast

to us Si et al.[33] apply their inceptionformer block in all stages. They also miss other optimizations which we describe in Section 4.1.

Our adaptation of MHSA is simple and is close to the traditional attention, setting aside overly complicated approaches to an already established concept. To the best of our knowledge nobody yet proposed this adaptation to the original Multi-Head Self-Attention [40].

3. Execution Time Analysis

In the following we will investigate the hardware-efficiency of convolutions in different configurations. A convolution or a block of convolutions is for example more hardware-efficient than another, when it has more MAC operations, but is at least similar fast. We will evaluate speed on desktop GPU and CPU.

In Section 4.2 we describe how we apply the insights gained from the following experiments.

3.1. Depthwise Convolutions

When the focus of a model is efficiency and mobile-friendly design, depthwise convolutions are a prominent alternative to standard convolutions [17, 34]. Standard convolutions are ungrouped convolutions (groups=1), while depthwise convolutions are grouped convolutions that have as many groups as input channels. While depthwise convolutions are efficient in terms of MACs, they can not translate latter completely to common hardware. Evaluating models by their MACs in architecture design automatically leads to inserting as many depthwise convolutions as possible, regardless of their effective speed-up. In order to concretize the disconnection of execution time and MACs as an efficiency measure, we carry out an experiment using a simplified toy architecture. We examine the effect of using depthwise convolutions on GPU throughput, GPU latency and CPU latency (see Table 1). We created three models with only depthwise convolutions (#2,#4,#6) and three with only standard ones (#1,#3,#5). Even though model #1 and #2 in Table 1 have a similar GPU throughput and Latency, #1 (ungrouped) has 4× the amount of MACs. The same is true for model #3 and #4. Model #5 (ungrouped) even has 6× the amount of MACs, while having a slightly higher throughput. Regarding GPU latency however, the models with ungrouped convolutions (#1, #3 and #5) are slightly slower.

In summary, depthwise convolutions are not as hardware-efficient as standard convolutions. They do not give the expected speed-up which their low amount of MACs might suggest.

3.2. Fused vs. Unfused MBConv

The mobile inverted bottleneck block (MBConv) [32] has a long and successful history in efficient backbones

Model	Channel [C_0, C_1, C_2, C_3, C_4]	Depthwise	MACs (M)	GPU Throughput (images/s)	Latency CPU	Latency GPU
#1	[17, 34, 68, 136, 273]	✗	887	**5263**	**7.22**	0.26
#2	[30, 60, 120, 240, 480]	✓	207	5025	10.05	**0.24**
#3	[32, 65, 130, 260, 260]	✗	2734	2617	**15.36**	0.51
#4	[60,120,240,480,480]	✓	750	**2624**	20.20	**0.45**
#5	[32, 96, 193, 387, 387]	✗	5270	**1886**	**22.32**	0.74
#6	[60, 180, 360, 720, 720]	✓	809	1805	25.84	**0.65**

Table 1. Comparison of throughput and latency of depthwise and standard convolutions in relation to amount of MACs. The six toy models (#1-6) are divided into three groups by similar throughput. Each model has the same architecture and only differs in the number channels (see second column) and if its convolutions are depthwise or not (see third column). The table shows, that standard convolutions that feature many more MACs (4× to 6× the amount) can still be similarly fast as depthwise convolutions. Bold entries refer to the best value in each group.

[17, 32, 34] and is still used by many new approaches [2, 8, 35]. It consists of two pointwise convolutions (PWConv) and a depthwise (DWConv) in between (see Figure 5 for a depiction of it). The PWConvs increase and decrease the channel dimension by the attributed expansion factor. As we concluded however in Section 3.1, depthwise convolutions are not particularly hardware-efficient. With the goal of improving hardware-efficiency, we therefore remove the depthwise convolution by utilizing the fused MBConv [14], which fuses the first PWConv with the DWConv into a standard convolution. Even though the fused MBConv is more hardware-efficient (as it features no depthwise convolution), it also has a significantly higher amount of MACs, although it has one layer less. Depthwise convolutions scale better with high channel dimension, because its amount of computations is linear in regard to channel dimension, wherefore will compare the execution time of both blocks under different configurations.

In Figure 2 we compare fused and unfused MBConvs with an expansion factor of four and of six (first and second row). We omit the scenarios where channel numbers range from 16 to 64 and resolutions from 7 to 28, as they are not relevant for backbone architectures and GPU utilization is too low for the results to have significance. In both figures we depict the average execution time on GPU (left side) and amount of MACs (right side) of the fused MBConv divided by the unfused one. We do this for various configurations of resolution and input channel dimension. We measure average execution time with a batch size of 200 and run for 100 iterations, the same way we measure throughput in Section 5.1. It can be seen in Figure 2 that resolution and channel dimension have a big influence on the relative execution time and even though the fused MBConv always has more MACs (values over one), it is faster in many scenarios (values smaller than one in left part). Only for high number of channels (over 512) the relative execution time worsens and approaches the relative MACs again. We also evaluate the GPU latency in the same way in the supplementary mate-

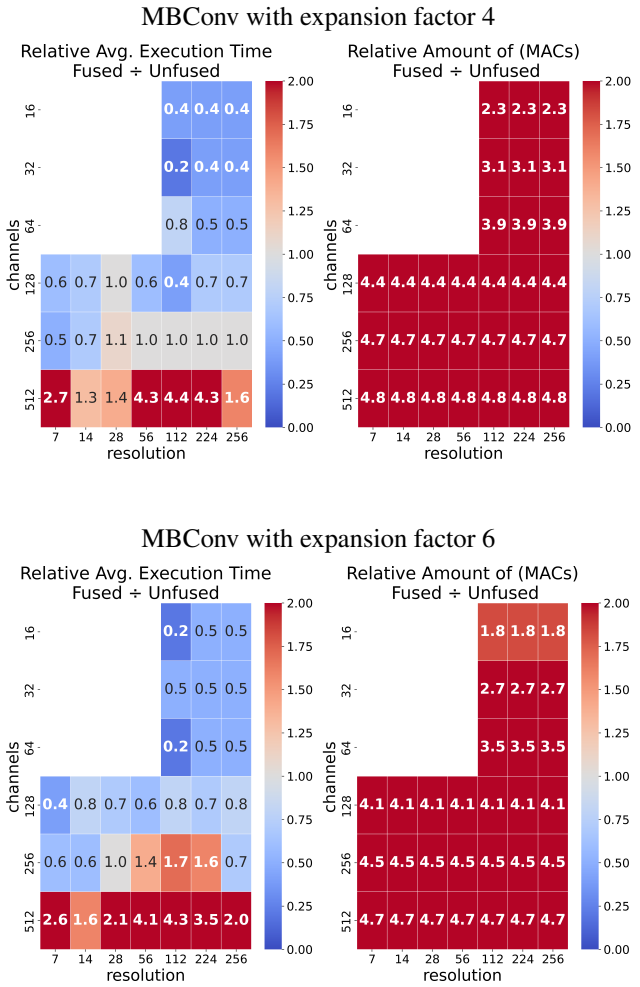

Figure 2. The left figures depict average execution time on GPU of the fused mobile inverted bottleneck (MBConv) relative to the unfused one, while the right figures depict the relative amount of MACs. The blue areas in the left figures correspond to configurations (number of channels and resolution) where fused MBConv is faster, while red corresponds to the opposite. For the right figures the red areas correspond to configurations where the fused MBConv has a higher amount of MACs. Bold and italic numbers refer to entries with a particularly unequal ratio. Even though the fused MBConv always has more MACs, it is faster for many configurations.

rial.

In Section 5.3 we additionally demonstrate this affect with our model LowFormer-B1.

3.3. High Resolution vs. High Channel

Operating resolution and the number of channels of a layer have a big influence on how hardware efficient the execution is. In Table 2 we compare different layer configurations by creating models that consist of 20 times the same layer stacked after another. We feature seven scenarios in Table 2, each contrasting two layers with the same

Scenario	Resolution (pixel)	Channels (#)	Relative Throughput	Relative Latency	Relative MACs
#1	224	24	0.3	2.7	1.0
	28	196	**3.3**	**0.37**	1.0
#2	224	48	0.5	1.88	1.0
	112	96	**1.9**	**0.53**	1.0
#3	56	96	0.5	1.5	1.0
	14	384	**2.2**	**0.67**	1.0
#4	112	96	0.6	1.91	1.0
	28	384	**1.8**	**0.52**	1.0
#5	224	96	0.5	2.16	1.0
	56	384	**1.9**	**0.46**	1.0
#6	112	24	0.3	2.3	1.0
	14	196	**3.3**	**0.44**	1.0
#7	224	48	0.5	1.15	1.0
	56	196	**2.0**	**0.87**	1.0

Table 2. Analysis of the impact of resolution on execution time relative to amount of MACs. Each scenario contains two configurations of convolutions (first row and second row in each scenario), that approximately feature the same amount of MACs (see 6th column), but strongly differ in operating resolution and number of channels. We set their throughput and latency on GPU in relation to each other(see 4th and 5th column). The table shows, that convolutions operating on a higher resolution tend to be slower than lower resolution convolutions with the same amount of MACs.

amount of MACs to one decimal place, but differing in channel dimension and operating resolution. To compare the hardware-efficiency of the layers, we measure GPU latency and throughput. The models only contain standard convolutions (ungrouped).

Table 2 clearly shows how a higher resolution results in a less hardware efficient execution, while a higher number of channels poses a minor problem. In scenario #1 for example the first convolution has a third of the throughput of the second one and almost thrice the latency. It operates on eight times the resolution, however features less channels and its MACs equal the second layer. The same effect also occurs with a smaller resolution difference, as scenario #2 shows, where the first convolution runs on twice the resolution, but still fails to execute its MACs as fast as the second one in terms of throughput and latency.

Operating on a high resolution can slow down a model more than the MACs might suggest, while reducing the operating resolution can lead to an considerably increased hardware-efficiency. We emphasize that model scaling should be more influenced by actual measured speed [34], as MACs can be misleading, especially when scaling models by higher input resolution [2, 47].

4. Methodology

In the following we describe our proposed lightweight adaptation of the original Multi-Head Self-Attention [40] and our hardware-efficient macro design. We also explain how both arose from the insights gained from the execution

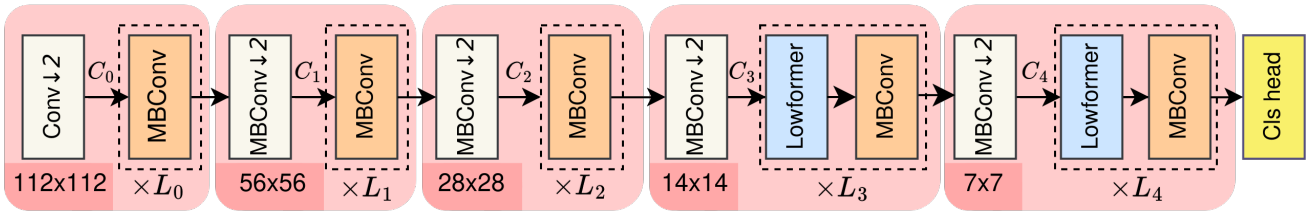

Figure 3. Architecture of LowFormer. The resolutions refer to a 224x224 sized input. LowFormer block can be seen in Figure 4. MBConv means the mobile inverted bottleneck block, Conv means convolution and Cls head refers to the image classification head. Specification of $C_0 - C_4$ and $L_0 - L_4$ can be found in Table 3.

time analysis in Section 3.

4.1. Lightweight Attention

In the last two stages we employ a lightweight adaptation of the original MHSA [40], depicted in Figure 4. We name it LowFormer Attention. In our adaptation the Scaled Dot-Product Attention (SDA) is encapsulated by two depthwise convolutions and two pointwise convolutions, which perform the input and output projections of the queries (Q), keys (K) and values (V).

Channel compression. During the input projection the channel dimension of Q, K and V is halved. After the SDA it is restored to the original channel dimension by the output projection.

Lower resolution. In Section 3.3 we learned how a high operating resolution slows down convolutions disproportionately. We transferred that insight to attention, wherefore convolutions (see Figure 4) also down- and upsample the resolution of the feature maps around the SDA in the forelast stage, such that the attention operation is executed on half the resolution. In Section 5.3 we show that this measure has a substantial effect on throughput and latency.

Due to these two adaptations all dimensions of the input to the SDA (channel dimension, height and width) are reduced, wherefore we connote LowFormer Attention with the attribute lightweight.

MLP following Attention. Following Vaswani et al.[40], we append layer normalization and a multi-layer perceptron (MLP) after the LowFormer Attention. We found that its effect on model accuracy is significant and Liu et al.[24] pointed out, the MLP is more hardware-efficient than the attention operation.

4.2. Macro Design

Our macro design is based on EfficientViT [2] and MobileViT [30]. We feature five stages and adapted the architecture mainly according to the following two recipes, that are fueled by the insights gained from the execution time analysis in Section 3. The whole architecture is depicted in Figure 3. We present five different versions of our architecture, namely B0-B3 (B0, B1, B1.5, B2, B3). Architecture details are listed in Table 3.

Less layers in the first stages. From the insights in Section 3.3 we conclude that a minimal amount of layers in the first stages is more hardware-efficient (see Table 3). It proved optimal to apply the reduction for the first three stages. Most computation is therefore concentrated in the last two stages, where for an input size of 224×224, the operating resolutions are 14×14 and 7×7.

Fusing Depthwise and Pointwise Convolutions. In Section 3.1 we showed that depthwise convolutions are not as hardware-efficient as standard convolution and in Section 3.2 we came to the conclusion that the fused MBConv (see Figure 5) can be faster than the unfused one. This effect diminishes however with increasing number of channels. We therefore fused the MBConv in our architecture, wherever the number of input channels reach at most 256, except for the stride 2 MBConv block in the last stage, which we fuse nevertheless (see Figure 3). We additionally fuse the depthwise and pointwise convolutions after the SDA in the LowFormer Attention (see Figure 4). We confirm the effect of this approach by reverting the fusion of the MBConv block for LowFormer-B1 in the ablation in Section 5.3.

Model	$\{L_0, L_1, L_2, L_3, L_4\}$	$\{C_0, C_1, C_2, C_3, C_4\}$
LowFormer-B0	$\{0, 0, 0, 3, 4\}$	$\{16, 32, 64, 128, 256\}$
LowFormer-B1	$\{0, 0, 0, 5, 5\}$	$\{16, 32, 64, 128, 256\}$
LowFormer-B1.5	$\{0, 0, 0, 6, 6\}$	$\{20, 40, 80, 160, 320\}$
LowFormer-B2	$\{0, 0, 0, 6, 6\}$	$\{24, 48, 96, 192, 384\}$
LowFormer-B3	$\{1, 1, 2, 6, 6\}$	$\{32, 64, 128, 256, 512\}$

Table 3. Specification of LowFormer architecture versions B0-B3. The number of layers ($L_0 - L_4$) and channels ($C_0 - C_4$) relates to Figure 3.

Model	Mobile GPU Latency (ms)	ARM CPU Latency (s)	Top-1 (%)
EfficientFormerV2-S0 [21]	26	1.201	73.7
MobileOne-S3 [2]	34	1.20	78.1
LowFormer-B0 (ours)	**20**	**0.835**	**78.4**
EfficientFormerV2-S1 [21]	35	1.388	77.9
ResNet-50 [16]	58	2.376	79.0
RepViT-M1.1 [41]	39	1.346	79.4
MobileOne-S4 [2]	45	1.889	79.4
LowFormer-B1 (ours)	**34**	**1.186**	**79.9**
EfficientFormerV2-S2 [21]	**54**	2.061	80.4
RepViT-M1.5 [41]	**54**	2.315	**81.2**
LowFormer-B1.5 (ours)	58	**1.970**	**81.2**

Table 4. Mobile GPU and ARM CPU Latency of backbones. Evaluation resolution is 224 × 224 and batch size is set to 1.

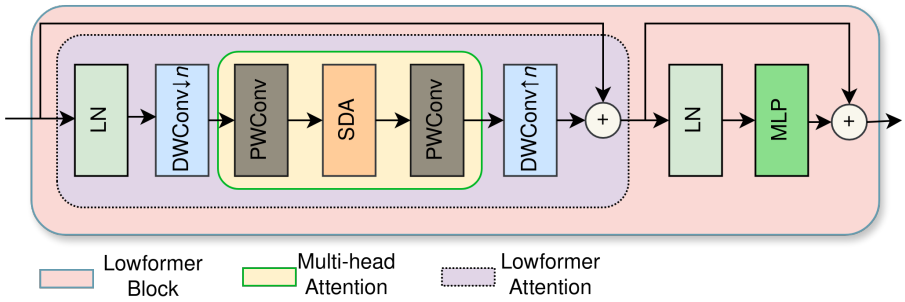

Figure 4. Lowformer block design. DWConv, PWConv, LN, MLP and SDA mean depthwise convolution, pointwise convolution, layer normalization, multi-layer perceptron and Scaled Dot-Product Attention respectively. In contrast to the traditional MHSA, we encapsulate the SDA with two depthwise convolutions (the second is a transposed depthwise convolution). The projections for MHSA are realized with pointwise convolutions. The $DW \downarrow_n$ means that the resolution is downscaled by the factor n and $DW \uparrow_n$ that it is upscaled by n.

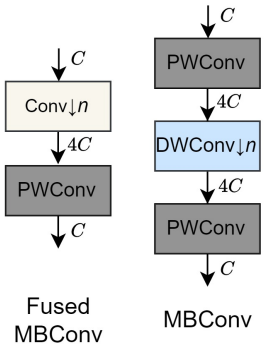

Figure 5. Structure of the fused and unfused MBConv block. C refers to the channel dimension. In Conv and DWConv the "↓n" refers to a potential stride. Both have an expansion factor of 4 in this figure.

5. Experiments

5.1. ImageNet-1K Classification

Settings. We conduct image classification experiments on ImageNet-1K [10], which includes 1.28M training and 50K validation images for 1000 categories. All models were trained from scratch using mostly the same setting as Cai et al.[2] and featuring an input resolution of 224. We also trained for a total of 320 epochs using AdamW [26] optimizer and a learning rate of 10^{-3}, however we use a batch size of 512. As learning rate scheduler we use cosine decay [25] and 20 warm-up epochs with a linear schedule. We also feature the multi-scale learning from Cai et al.[2]. We trained LowFormer-B3 with a batch size of 2400 and a base learning rate of 3×10^{-3}. For LowFormer-B2 we had a batch size of 850 and a base learning rate of 8.3×10^{-4}.

GPU Throughput and Latency. In Table 5 we measure speed by GPU throughput and latency. For GPU throughput we run for 100 iterations on a Nvidia A40 graphic card with a batch size of 200 and take the median time per input image to calculate the total throughput. To measure GPU latency we run for 400 iterations on a Nvidia Titan RTX, with a batch size of 16. For the latter we also compile all models to TorchScript code and optimize them for inference [1]. We feature 5 warm-up iterations for latency and throughput measurement.

Results. For a better comparison between competing approaches we include a version of LowFormer-B1 and LowFormer-B3 that are evaluated on a different resolution (see Table 5). The variants of LowFormer outperform all other approaches (see Table 5) with a similar or lower GPU throughput in regard to top-1 accuracy. LowFormer-B1$_{r256}$, tested on resolution 256, has a 0.8% higher top-1 accuracy than EfficientViT-B1 [2] and a 23% higher throughput. It also similarly outperforms RepViT-M1.1 [41], but has a 73% higher throughput and a 21% lower latency. LowFormer-B1, tested on resolution 224, only achieves a 0.44 % increase in accuracy compared to SHViT-S4$_{r256}$ [47], with a similar throughput. The SHViT architecture however only increases their model complexity in an inefficient way (see Section 3.3) by increasing input resolution, wherefore their best model SHViT-S4$_{r512}$ (tested on resolution 512) scores 1.64 % worse in top-1 accuracy than LowFormer-B3 with a similar throughput. The hardware-efficiency of our design and the inconsistent relationship between amount of MACs and execution time becomes especially apparent, when comparing LowFormer-B3 to GhostNetV2x1.0 [36]. Former has 3× the throughput, 20% lower latency, 36 × the amount of MACs and scores 8.3% better in top-1 accuracy.

Resolution Scaling. In Figure 6 we investigate for LowFormer-B1 and other approaches the impact of increasing the input resolution on GPU latency. LowFormer-B1 outperforms depicted models in top-1 accuracy on ImageNet-1K and at the same time remains considerably faster, independent of the input resolution. In the supplementary we further perform the same evaluation with a batch size of 200 to estimate change in throughput.

Mobile GPU and CPU Latency. In Table 4 we also verify the efficiency of our model on mobile GPU (ARM Mali-G76 MP12) and ARM CPU (ARM Cortex A53). We run the models with a batch size of 1 and a resolution of 224×224. For the ARM CPU we ran for 30 iterations, while for the mobile GPU we ran for 10000 iterations, with 1000

[1] https://pytorch.org/docs/stable/generated/torch.jit.optimize_for_inference.html

Model	Venue	Params (M)	MACs (M)	GPU Throughput (images/s)	GPU Latency (ms)	Resolution (pixel)	Top-1 (%)
MobileViG-Ti [31]	CVPRW 2023	5.3	661	2500	0.39	224	75.7
PVTv2-B0 [44]	CVM 2022	3.4	566	2164	0.50	224	70.5
EdgeViT-XXS [6]	ECCV 2022	4.1	551	2816	0.41	224	74.4
GhostNetV2 x1.0 [36]	NeurIPS 2022	6.2	183	375	1.93	224	75.3
FastViT-T8 [38]	CVPR 2023	3.6	690	1694	0.54	256	75.6
EfficientMod-xxs [29]	ICLR 2024	4.7	579	2857	0.39	224	76.0
EfficientViT-M5 [24]	CVPR 2023	12.4	521	5681	0.30	224	77.1
RepViT-M0.9 [41]	CVPR 2024	5.1	816	2512	0.47	224	77.4
SHViT-S3 [47]	CVPR 2024	14.3	601	5780	0.28	224	77.4
MobileOne-S2 [39]	CVPR 2023	7.8	1298	2967	0.35	224	77.4
LowFormer-B0 (ours)		14.1	944	**5988**	0.30	224	**78.4**
EdgeViT-XS [6]	ECCV 2022	6.8	1127	2127	0.56	224	77.5
MobileOne-S3 [39]	CVPR 2023	10.1	1895	2433	0.48	224	78.1
MobileViG-S [31]	CVPRW 2023	7.3	983	1724	0.57	224	78.2
EfficientMod-xs [29]	ICLR 2024	6.6	773	2352	0.50	224	78.3
PVTv2-B1 [44]	CVM 2022	13.1	2108	1228	0.98	224	78.7
EfficientViT-B1 [2]	ICCV 2023	9.1	519	2739	0.44	224	79.4
SHViT-S4 [47]	CVPR 2024	16.5	986	**4255**	0.35	256	79.4
LowFormer-B1 (ours)		17.9	1410	4237	0.43	224	**79.9**
FAT-B0 [12]	NeurIPS 2023	4.5	754	1620	0.70	224	77.6
MobileOne-S4 [39]	CVPR 2023	14.8	297	1550	0.82	224	79.4
RepViT-M1.1 [41]	CVPR 2024	8.2	1338	1941	0.61	224	79.4
EfficientViT-M5 [24]	CVPR 2023	12.4	1494	2247	0.51	384	79.8
LowFormer-B1 (ours)		17.9	1843	**3378**	0.48	256	**80.2**
FAT-B1 [12]	NeurIPS 2023	7.8	1243	1204	0.96	224	80.1
FastViT-SA12 [38]	CVPR 2023	10.9	1943	1075	0.95	256	80.6
SHViT-S4 [47]	CVPR 2024	16.5	2224	1785	0.62	384	81.0
EdgeViT-S [6]	ECCV 2022	11.1	1910	1449	0.85	224	81.0
LowFormer-B1.5 (ours)		33.9	2573	**2739**	0.66	224	**81.2**
EfficientViT-M5 [24]	CVPR 2023	12.4	2799	1197	0.92	512	80.8
EfficientMod-s [29]	ICLR 2024	12.9	1402	1381	0.83	224	81.0
RepViT-M1.5 [41]	CVPR 2024	14.0	2276	1146	1.03	224	81.2
FFNet-1 [48]	arXiv 2024	13.8	3000	1090	0.96	256	81.3
LowFormer-B2 (ours)		45.0	3689	**2227**	0.88	224	**81.6**
SHViT-S4 [47]	CVPR 2024	16.5	3971	1160	1.04	512	82.0
EfficientViT-B2 [2]	ICCV 2023	15.0	1584	1298	0.92	224	82.1
RepViT-M2.3 [41]	CVPR 2024	22.9	4520	642	1.84	224	82.5
FastViT-SA24 [38]	CVPR 2023	10.9	3769	606	1.70	256	82.6
MobileViG-B [31]	CVPRW 2023	26.7	2792	869	1.24	224	82.6
LowFormer-B3 (ours)		57.1	4479	**1562**	1.24	192	**82.7**
FFNet-2 [48]	arXiv 2024	26.9	6060	496	2.12	256	82.9
iFormer-S [33]	NeurIPS 2022	19.9	4825	555	2.11	224	83.4
EfficientViT-B3 [2]	ICCV 2023	39.0	3953	666	1.88	224	83.4
FAT-B3 [12]	NeurIPS 2023	29.0	4631	402	2.82	224	83.6
FastViT-SA36 [38]	CVPR 2023	30.4	5595	429	2.00	256	83.6
LowFormer-B3 (ours)		57.1	6098	**1162**	1.55	224	**83.6**

Table 5. Performance on ImageNet-1K validation set. Neither distillation nor pretraining is used for fair comparison. The table is divided in different complexity classes, determined by throughput. Results in **bold** are the best results for each complexity class, while underlined results refer to the second best.

warm-up iterations. Even though our architecture is not specifically optimized for edge application, LowFormer-B1 for examples scores 0.5% better in top-1 accuracy than MobileOne-S4 [39], who has a 32% higher mobile GPU latency and a 59% higher ARM CPU latency.

5.2. Object Detection & Semantic Segmentation

We utilize the pretrained backbones and apply them in object detection and semantic segmentation. We train and evaluate on COCO 2017 [22] and ADE20K [49] respectively using mmdetection [4] and mmsegmentation [7]. For object detection we take the RetinaNet framework [23], while for Semantic Segmentation we plug our backbone into the Semantic FPN [18]. For backbone GPU throughput and latency measurement we follow the protocol in Section 5.1, but evaluate on input resolution of 512×512.

Object Detection. We train the models for 12 epochs (1x schedule), following [2, 12]. Regarding results, LowFormer-B2 for example outperforms FAT-B0 [12] by **+1.0** AP, while having 93% higher backbone throughput on

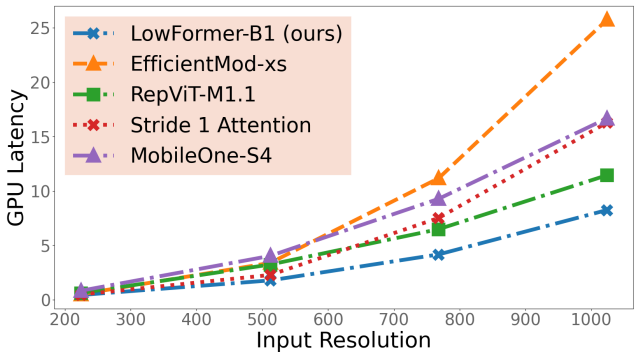

Figure 6. Impact of input resolution on GPU Latency for LowFormer-B1 (ours), LowFormer-B1 without downsampling in LowFormer Attention (see Section 5.3), EfficientMod-xs [29], FastViT-T8 [38] and RepViT-M1.1 [41].

Backbone	Throughput (images/s)	Latency (ms)	AP (%)	AP_{50} (%)	AP_{75} (%)	AP_s (%)	AP_m (%)	AP_l (%)
MobileNetV3 [17]	862	0.97	29.9	49.3	30.8	14.9	33.3	41.1
EfficientViT-M4 [24]	**1700**	0.58	32.7	52.2	34.1	17.6	35.3	46.0
PVTv2-B0 [44]	355	3.01	37.2	57.2	39.5	**23.1**	40.4	49.7
LowFormer-B0 (ours)	1190	1.24	**38.6**	**59.1**	**40.9**	21.8	**41.8**	**51.7**
EdgeViT-XXS [6]	518	2.18	38.7	59.0	41.0	**22.4**	42.0	51.6
LowFormer-B1 (ours)	**840**	1.77	**39.4**	**59.8**	**41.7**	**22.4**	**42.9**	**52.4**
FAT-B0 [12]	232	4.58	40.4	61.6	42.7	24.0	44.3	53.1
EdgeViT-XS [6]	400	2.89	40.6	61.3	43.3	25.2	43.9	54.6
PVTv2-B1 [44]	215	5.22	41.2	61.9	43.9	**25.4**	44.5	54.3
LowFormer-B2 (ours)	**450**	4.42	**41.4**	**62.2**	**44.1**	24.5	**45.1**	**55.5**
FAT-B1 [12]	174	6.32	42.5	64.0	45.1	26.9	46.0	**56.7**
LowFormer-B3 (ours)	**245**	7.40	**43.1**	**64.5**	**45.9**	**27.1**	**47.1**	**56.7**

Table 6. Comparison results on object detection on COCO 2017 [22] using RetinaNet [23] head. Backbone throughput and latency are measured under resolution of 512×512 and on GPU.

resolution 512×512 (see Table 6).

Semantic Segmentation. For semantic segmentation we train the models for 40K iterations with a batch size of 32, following [12, 29, 38, 41]. We use AdamW optimizer [26], cosine annealing for the learning rate [25] with a base learning rate of 2×10^{-3} and 1K warm-up steps with linear increase. LowFormer-B1 for example has 2.4× the throughput and a 20% lower latency than EfficientFormerV2-S2[21], but achieves **+0.4** mIoU when plugged into Semantic FPN (see Table 7).

5.3. Ablation Study

In Table 8 we ablate our model design decisions. We revert a singular design decision of LowFormer-B1 to demonstrate the impact of that change on accuracy, throughput and latency. The featured ablations are the following:

- We replace all fused MBConv blocks with the unfused version.
- We remove the LowFormer block and add an additional MBConv to each stage, such that the ablated model has the same throughput as the baseline (called "attention removed" in Table 8).

Backbone	GPU Throughput (images/s)	GPU Latency (ms)	mIoU (%)
ResNet50 [16]	271	3.32	36.7
PVTv2-B0 [44]	355	3.01	37.2
FastViT-SA12 [38]	265	3.77	38.0
EdgeViT-XXS [6]	518	2.18	39.7
LowFormer-B1 (ours)	**840**	1.77	**39.7**
RepViT-M1.1 [41]	404	2.87	40.6
FastViT-SA24 [38]	151	6.88	41.0
EdgeViT-XS [6]	400	2.89	41.4
FAT-B0 [12]	232	4.58	41.5
EfficientFormerV2-S2 [21]	182	5.58	42.4
PVTv2-B1 [44]	215	5.22	42.5
LowFormer-B2 (ours)	**450**	4.42	**42.8**
FAT-B1 [12]	174	6.32	42.9
RepViT-M1.5 [41]	238	5.08	43.6
FastViT-MA36 [38]	86	13.18	44.6
LowFormer-B3 (ours)	**245**	7.40	**44.6**

Table 7. Results on semantic segmentation, using Semantic FPN [18]. Backbone throughput and latency are measured under resolution of 512×512.

Model version	Params (M)	MACs (M)	GPU Throughput (images/s)	GPU Latency (ms)	Top-1 (%)
unfused MBConv	12.4	716	3558 (-16%)	0.43	79.1 (-0.8)
attention removed	14.46	1643	4098 (-3%)	0.41	79.7 (-0.2)
relulinear att	14.15	1210	3367 (-20%)	0.49	79.6 (-0.3)
high-res attention	17.65	1494	3759 (-11%)	0.47	**79.9** (-0.0)
Baseline (B1)	17.94	1410	**4237**	0.43	**79.9**

Table 8. Ablation study of LowFormer-B1, featuring singular changes to the original model and putting them in relation to the original model. Bold entries mark the best in its column.

- We replace our LowFormer Attention with ReLU linear attention from Cai et al.[2] in order to compare our attention approach with other recent adaptations.
- We omit the downscaling of the feature maps for the LowFormer Attention.

Discussion. Replacing the fused MBConv with the unfused one results in a 0.8% lower top-1 accuracy, while its GPU throughput is also 16% lower (see Table 8). As we can see, next to an improved throughput, fusing the MBConv can increase performance significantly.
On the other side, replacing the LowFormer block with convolutional layers in every stage only results in a smaller drop in top-1 accuracy. Because of the additional layers however, the potential to upscale the depth of the architecture is reduced, as the more layers a stage has, the less effective they deliberately become (see Tan and Le[34]).
When we remove the downsampling for the LowFormer Attention, top-1 accuracy stays the same, but GPU throughput and latency worsen. In Figure 6 we can see that for higher input resolutions the latency difference multiplies. For input resolution 1024×1024 for example, latency is increased by 70%.

6. Conclusion

We have shown how a high resolution and depthwise convolutions negatively impact hardware-efficiency and proposed a recipe on how depthwise convolutions can be replaced in an architecture. We also proposed a simple lightweight attention and demonstrated that letting it operate on a lower resolution does not reduce accuracy, but leads to a considerably faster execution, especially when the input resolution of the model is increased. Our hardware-efficient macro and micro design yields significant speed-ups compared to previous approaches. We further proved the applicability of our backbone architecture to object detection and semantic segmentation.

Acknowledgements

This work was funded by European Union-NextGenerationEU under Project PRIN 2022 EXTRA EYE and Project PRIN 2022 PNRR TEAM.

References

[1] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, and Judy Hoffman. Hydra attention: Efficient attention with many heads. In *European Conference on Computer Vision*, pages 35–49. Springer, 2022. 2

[2] Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17302–17313, 2023. 1, 2, 3, 4, 5, 6, 7, 8

[3] 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*, pages 12021–12031, 2023. 2

[4] 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. 7

[5] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-former: Bridging mobilenet and transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5270–5279, 2022. 2

[6] Zekai Chen, Fangtian Zhong, Qi Luo, Xiao Zhang, and Yanwei Zheng. Edgevit: Efficient visual modeling for edge computing. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 393–405. Springer, 2022. 2, 7, 8

[7] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 7

[8] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in neural information processing systems*, 34:3965–3977, 2021. 3

[9] Mostafa Dehghani, Anurag Arnab, Lucas Beyer, Ashish Vaswani, and Yi Tay. The efficiency misnomer. *arXiv preprint arXiv:2110.12894*, 2021. 2

[10] 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, 6

[11] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6824–6835, 2021. 2

[12] Qihang Fan, Huaibo Huang, Xiaoqiang Zhou, and Ran He. Lightweight vision transformer with bidirectional interaction. *Advances in Neural Information Processing Systems*, 36, 2023. 2, 7, 8

[13] Micah Goldblum, Hossein Souri, Renkun Ni, Manli Shu, Viraj Prabhu, Gowthami Somepalli, Prithvijit Chattopadhyay, Mark Ibrahim, Adrien Bardes, Judy Hoffman, et al. Battle of the backbones: A large-scale comparison of pretrained models across computer vision tasks. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[14] Suyog Gupta and Mingxing Tan. Efficientnet-edgetpu: Creating accelerator-optimized neural networks with automl. *https://ai.googleblog.com/2019/08/efficientnetedgetpu-creating.html*, 2019. 3

[15] 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. 2

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5, 8

[17] 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. 2, 3, 8

[18] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6399–6408, 2019. 2, 7, 8

[19] Youngwan Lee, Joong-won Hwang, Sangrok Lee, Yuseok Bae, and Jongyoul Park. An energy and gpu-computation efficient backbone network for real-time object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019. 1

[20] 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. 2

[21] 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*, pages 16889–16900, 2023. 5, 8

[22] 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. 7, 8

[23] 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. 2, 7, 8

[24] 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. 1, 2, 5, 7, 8

[25] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6, 8

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6, 8

[27] 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. 2

[28] 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. 2

[29] Xu Ma, Xiyang Dai, Jianwei Yang, Bin Xiao, Yinpeng Chen, Yun Fu, and Lu Yuan. Efficient modulation for vision networks. *arXiv preprint arXiv:2403.19963*, 2024. 7, 8

[30] Sachin Mehta and Mohammad Rastegari. Mobilevit: lightweight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021. 1, 5

[31] Mustafa Munir, William Avery, and Radu Marculescu. Mobilevig: Graph-based sparse attention for mobile vision applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2211–2219, 2023. 2, 7

[32] 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. 3

[33] Chenyang Si, Weihao Yu, Pan Zhou, Yichen Zhou, Xinchao Wang, and Shuicheng Yan. Inception transformer. *Advances in Neural Information Processing Systems*, 35:23495–23509, 2022. 2, 3, 7

[34] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 1, 2, 3, 4, 8

[35] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021. 3

[36] Yehui Tang, Kai Han, Jianyuan Guo, Chang Xu, Chao Xu, and Yunhe Wang. Ghostnetv2: Enhance cheap operation with long-range attention. *Advances in Neural Information Processing Systems*, 35:9969–9982, 2022. 2, 6, 7

[37] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *European conference on computer vision*, pages 459–479. Springer, 2022. 2

[38] 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*, pages 5785–5795, 2023. 2, 7, 8

[39] 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. 1, 2, 7

[40] 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. 2, 3, 4, 5

[41] Ao Wang, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Repvit: Revisiting mobile cnn from vit perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15909–15920, 2024. 5, 6, 7, 8

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

[43] 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. 2

[44] 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, 8

[45] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers, 2021. 2

[46] 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

[47] Seokju Yun and Youngmin Ro. Shvit: Single-head vision transformer with memory efficient macro design. *arXiv preprint arXiv:2401.16456*, 2024. 1, 2, 4, 6, 7

[48] Seokju Yun, Dongheon Lee, and Youngmin Ro. Metamixer is all you need. *arXiv preprint arXiv:2406.02021*, 2024. 7

[49] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 7