EdgeNeXt: Efficiently Amalgamated CNN-Transformer Architecture for Mobile Vision Applications

Muhammad Maaz[1*], Abdelrahman Shaker[1*], Hisham Cholakkal[1], Salman Khan[1,2], Syed Waqas Zamir[3], Rao Muhammad Anwer[1,4], and Fahad Shahbaz Khan[1,5]

[1]Mohamed bin Zayed University of AI   [2]Australian National University
[3]Inception Institute of Artificial Intelligence (IIAI)   [4]Aalto University
[5]Linköping University

Abstract. In the pursuit of achieving ever-increasing accuracy, large and complex neural networks are usually developed. Such models demand high computational resources and therefore cannot be deployed on edge devices. It is of great interest to build resource-efficient general purpose networks due to their usefulness in several application areas. In this work, we strive to effectively combine the strengths of both CNN and Transformer models and propose a new efficient hybrid architecture EdgeNeXt. Specifically in EdgeNeXt, we introduce split depth-wise transpose attention (STDA) encoder that splits input tensors into multiple channel groups and utilizes depth-wise convolution along with self-attention across channel dimensions to implicitly increase the receptive field and encode multi-scale features. Our extensive experiments on classification, detection and segmentation tasks, reveal the merits of the proposed approach, outperforming state-of-the-art methods with comparatively lower compute requirements. Our EdgeNeXt model with 1.3M parameters achieves 71.2% top-1 accuracy on ImageNet-1K, outperforming MobileViT with an absolute gain of 2.2% with 28% reduction in FLOPs. Further, our EdgeNeXt model with 5.6M parameters achieves 79.4% top-1 accuracy on ImageNet-1K. The code and models are available at https://t.ly/_Vu9.

Keywords: Edge devices, Hybrid model, Convolutional neural network, Self-attention, Transformers, Image classification, Object detection and Segmentation

1 Introduction

Convolutional neural networks (CNNs) and the recently introduced vision transformers (ViTs) have significantly advanced the state-of-the-art in several mainstream computer vision tasks, including object recognition, detection and segmentation [37,20]. The general trend is to make the network architectures more deeper and sophisticated in the pursuit of ever-increasing accuracy. While striving for higher accuracy, most existing CNN and ViT-based architectures ignore the aspect of computational efficiency (*i.e.*, model size and speed) which is crucial to operating on resource-constrained devices such as mobile platforms. In many real-world applications *e.g.*, robotics and self-driving cars,

---

[*]Equal contribution

the recognition process is desired to be both accurate and have low latency on resource-constrained mobile platforms.

Most existing approaches typically utilize carefully designed efficient variants of convolutions to achieve a trade-off between speed and accuracy on resource-constrained mobile platforms [19,28,36]. Other than these approaches, few existing works [16,39] employ hardware-aware neural architecture search (NAS) to build low latency accurate models for mobile devices. While being easy to train and efficient in encoding local image details, these aforementioned light-weight CNNs do not explicitly model global interactions between pixels.

The introduction of self-attention in vision transformers (ViTs) [8] has made it possible to explicitly model this global interaction, however, this typically comes at the cost of slow inference because of the self-attention computation [24]. This becomes an important challenge for designing a lightweight ViT variant for mobile vision applications.

The majority of the existing works employ CNN-based designs in developing efficient models. However, the convolution operation in CNNs inherits two main limitations: First, it has local receptive field and thereby unable to model global context; Second, the learned weights are stationary at inference times, making CNNs inflexible to adapt to the input content. While both of these issues can be alleviated with Transformers, they are typically compute intensive. Few recent works [46,29] have
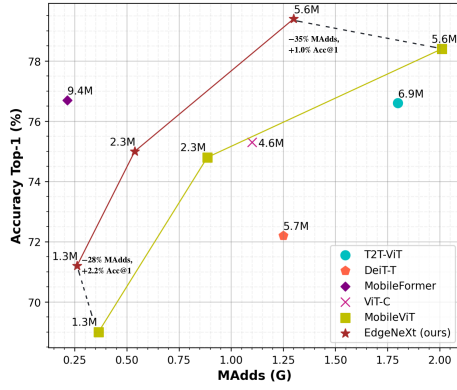

Fig. 1: Comparison of our proposed EdgeNeXt models with SOTA ViTs and hybrid architecture designs. The x-axis shows the multiplication-addition (MAdd) operations and y-axis displays the top-1 ImageNet-1K classification accuracy. The number of parameters are mentioned for each corresponding point in the graph. Our EdgeNeXt shows better compute (parameters and MAdds) versus accuracy trade-off compared to recent approaches.

investigated designing lightweight architectures for mobile vision tasks by combining the strengths of CNNs and ViTs. However, these approaches mainly focus on optimizing the parameters and incur higher multiply-adds (MAdds) operations which restricts high-speed inference on mobile devices. The MAdds are higher since the complexity of the attention block is quadratic with respect to the input size [29]. This becomes further problematic due to multiple attention blocks in the network architecture. Here, we argue that the model size, parameters, and MAdds are all desired to be small with respect to the resource-constrained devices when designing a unified mobile architecture that effectively combines the complementary advantages of CNNs and ViTs (see Fig. 1).

Contributions. We propose a new light-weight architecture, named *EdgeNeXt*, that is efficient in terms of model size, parameters *and* MAdds, while being superior in accuracy on mobile vision tasks. Specifically, we introduce split depth-wise transpose attention (SDTA) encoder that effectively learns both local and global representations

to address the issue of limited receptive fields in CNNs without increasing the number of parameters and MAdd operations. Our proposed architecture shows favorable performance in terms of both accuracy *and* latency compared to state-of-the-art mobile networks on various tasks including image classification, object detection, and semantic segmentation. Our EdgeNeXt backbone with 5.6M parameters and 1.3G MAdds achieves 79.4% top-1 ImageNet-1K classification accuracy which is superior to its recently introduced MobileViT counterpart [29], while requiring 35% less MAdds. For object detection and semantic segmentation tasks, the proposed EdgeNeXt achieves higher mAP and mIOU with fewer MAdds and a comparable number of parameters, compared to all the published lightweight models in literature.

2   Related Work

In recent years, designing lightweight hardware-efficient convolutional neural networks for mobile vision tasks has been well studied in literature. The current methods focus on designing efficient versions of convolutions for low-powered edge devices [19,17]. Among these methods, MobileNet [17] is the most widely used architecture which employs depth-wise separable convolutions [5]. On the other hand, ShuffleNet [47] uses channel shuffling and low-cost group convolutions. MobileNetV2 [36] introduces inverted residual block with linear bottleneck, achieving promising performance on various vision tasks. ESPNetv2 [31] utilizes depth-wise dilated convolutions to increase the receptive field of the network without increasing the network complexity. The hardware-aware neural architecture search (NAS) has also been explored to find a better trade-off between speed and accuracy on mobile devices [16,39]. Although these CCNs are faster to train and infer on mobile devices, they lack global interaction between pixels which limits their accuracy.

Recently, Desovitskiy *et al.* [8] introduces a vision transformer architecture based on the self-attention mechanism [41] for vision tasks. Their proposed architecture utilizes large-scale pre-training data (e.g. JFT-300M), extensive data augmentations, and a longer training schedule to achieve competitive performance. Later, DeiT [40] proposes to integrate distillation token in this architecture and only employ training on ImageNet-1K [35] dataset. Since then, several variants of ViTs and hybrid architectures are proposed in the literature, adding image-specific inductive bias to ViTs for obtaining improved performance on different vision tasks [38,9,42,48,11].

ViT models achieve competitive results for several visual recognition tasks [8,24]. However, it is difficult to deploy these models on resource-constrained edge devices because of the high computational cost of the multi-headed self-attention (MHA). There has been recent work on designing lightweight hybrid networks for mobile vision tasks that combine the advantages of CNNs and transformers. MobileFormer [4] employs parallel branches of MobileNetV2 [36] and ViTs [8] with a bridge connecting both branches for local-global interaction. Mehta *et al.* [29] consider transformers as convolution and propose a MobileViT block for local-global image context fusion. Their approach achieves superior performance on image classification surpassing previous light-weight CNNs and ViTs using a similar parameter budget.

Although MobileViT [29] mainly focuses on optimizing parameters and latency, MHA is still the main efficiency bottleneck in this model, especially for the number of MAdds and the inference time on edge devices. The complexity of MHA in Mobile-ViT is quadratic with respect to the input size, which is the main efficiency bottleneck given their existing nine attention blocks in MobileViT-S model. In this work, we strive to design a new light-weight architecture for mobile devices that is efficient in terms of both parameters and MAdds, while being superior in accuracy on mobile vision tasks. Our proposed architecture, EdgeNeXt, is built on the recently introduced CNN method, ConvNeXt [25], which modernizes the ResNet [14] architecture following the ViT design choices. Within our EdgeNeXt, we introduce an SDTA block that combines depth-wise convolutions with adaptive kernel sizes along with transpose attention in an efficient manner, obtaining an optimal accuracy-speed trade-off.

3 EdgeNeXt

The main objective of this work is to develop a lightweight hybrid design that effectively fuses the merits of ViTs and CNNs for low-powered edge devices. The computational overhead in ViTs (e.g., MobileViT [29]) is mainly due to the self-attention operation. In contrast to MobileViT, the attention block in our model has linear complexity with respect to the input spatial dimension of $\mathcal{O}(Nd^2)$, where N is the number of patches, and d is the feature/channel dimension. The self-attention operation in our model is applied across channel dimensions instead of the spatial dimension. Furthermore, we demonstrate that with a much lower number of attention blocks (3 versus 9 in Mobile-ViT), we can surpass their performance mark. In this way, the proposed framework can model global representations with a limited number of MAdds which is a fundamental criterion to ensure low-latency inference on edge devices. To motivate our proposed architecture, we present two desirable properties.

a) Encoding the global information efficiently. The intrinsic characteristic of self-attention to learn global representations is crucial for vision tasks. To inherit this advantage efficiently, we use cross-covariance attention to incorporate the attention operation across the feature channel dimension instead of the spatial dimension within a relatively small number of network blocks. This reduces the complexity of the original self-attention operation from quadratic to linear in terms of number of tokens and implicitly encodes the global information effectively.

b) Adaptive kernel sizes. Large-kernel convolutions are known to be computationally expensive since the number of parameters and FLOPs quadratically increases as the kernel size grows. Although a larger kernel size is helpful to increase the receptive field, using such large kernels across the whole network hierarchy is expensive and sub-optimal. We propose an adaptive kernel sizes mechanism to reduce this complexity and capture different levels of features in the network. Inspired by the hierarchy of the CNNs, we use smaller kernels at the early stages, while larger kernels at the latter stages in the convolution encoder blocks. This design choice is optimal as early stages in CNN usually capture low-level features and smaller kernels are suitable for this purpose. However, in later stages of the network, large convolutional kernels are required to capture high-level features [45]. We explain our architectural details next.
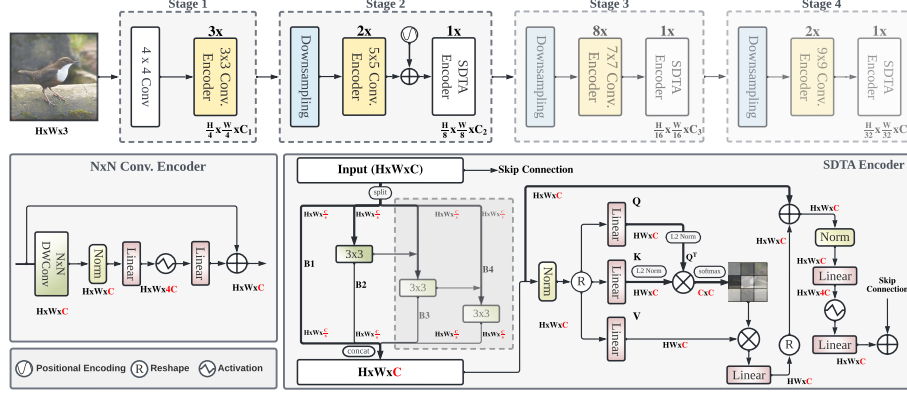
Fig. 2: **Top Row:** The overall architecture of our framework is a stage-wise design. Here, the first stage downsamples the input image to $1/4^{th}$ resolution using 4×4 strided convolution followed by three 3×3 Convolution (Conv.) encoders. In stages 2-4, 2×2 strided convolutions are used for downsampling at the start, followed by $N \times N$ Convolution and the Split depth-wise Transpose Attention (SDTA) encoders. **Bottom Row:** We present the design of the Conv. encoder (Left) and the SDTA encoder (right). The Conv. encoder uses $N \times N$ depth-wise convolutions for spatial mixing followed by two pointwise convolutions for channel mixing. The SDTA Encoder splits the input tensor into B channel groups and applies 3×3 depth-wise convolutions for multi-scale spatial mixing. The skip connections between branches increase the overall receptive field of the network. The branches $B3$ and $B4$ are progressively activated in stages 3 and 4, increasing the overall receptive field in the deeper layers of the network. Within the proposed SDTA, we utilize Transpose Attention followed by a light-weight MLP, that applies attention to feature channels and has linear complexity with respect to the input image.

Overall Architecture. Fig. 2 illustrates an overview of the proposed EdgeNeXt architecture. The main ingredients are two-fold: **(1)** adaptive $N \times N$ Conv. encoder, and **(2)** split depth-wise transpose attention (SDTA) encoder. Our EdgeNeXt architecture builds on the design principles of ConvNeXt [25] and extracts hierarchical features at four different scales across the four stages. The input image of size $H \times W \times 3$ is passed through a patchify stem layer at the beginning of the network, implemented using a 4×4 non-overlapping convolution followed by a layer norm, which results in $\frac{H}{4} \times \frac{W}{4} \times C1$ feature maps. Then, the output is passed to 3×3 Conv. encoder to extract local features. The second stage begins with a downsampling layer implemented using 2×2 strided convolution that reduces the spatial sizes by half and increases the channels, resulting in $\frac{H}{8} \times \frac{W}{8} \times C2$ feature maps, followed by two consecutive 5×5 Conv. encoders. Positional Encoding (PE) is also added before the SDTA block in the second stage only. We observe that PE is sensitive for dense prediction tasks (e.g., object detection and segmentation) as well as adding it in all stages increases the latency of the network. Hence, we add it only once in the network to encode the spatial location information. The output feature maps are further passed to the third and fourth stages, to generate $\frac{H}{16} \times \frac{W}{16} \times C3$ and $\frac{H}{32} \times \frac{W}{32} \times C4$ dimensional features, respectively.

Convolution Encoder. This block consists of depth-wise separable convolution with adaptive kernel sizes. We can define it by two separate layers: **(1)** depth-wise convolution with adaptive $N \times N$ kernels. We use k = 3, 5, 7, and 9 for stages 1, 2, 3, and 4,

respectively. Then, **(2)** two point-wise convolution layers are used to enrich the local representation alongside standard Layer Normalization [2] (LN) and Gaussian Error Linear Unit [15] (GELU) activation for non-linear feature mapping. Finally, a skip connection is added to make information flow across the network hierarchy. This block is similar to the ConvNeXt block but the kernel sizes are dynamic and vary depending on the stage. We observe that adaptive kernel sizes in Conv. encoder perform better compared to static kernel sizes (Table 8). The Conv. encoder can be represented as follows:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + Linear_G\big(Linear(LN(Dw(\boldsymbol{x}_i)))\big), \tag{1}$$

where $\boldsymbol{x}_i$ denotes the input feature maps of shape $H \times W \times C$, $Linear_G$ is a point-wise convolution layer followed by GELU, Dw is $k \times k$ depth-wise convolution, LN is a normalization layer, and $\boldsymbol{x}_{i+1}$ denotes the output feature maps of the Conv. encoder.

SDTA Encoder. There are two main components in the proposed split depth-wise transpose attention (SDTA) encoder. The first component strives to learn an adaptive multi-scale feature representation by encoding various spatial levels within the input image and the second part implicitly encodes global image representations. The first part of our encoder is inspired by Res2Net [12] where we adopt a multi-scale processing approach by developing hierarchical representation into a single block. This makes the spatial receptive field of the output feature representation more flexible and adaptive. Different from Res2Net, the first block in our SDTA encoder does not use the 1×1 pointwise convolution layers to ensure a lightweight network with a constrained number of parameters and MAdds. Also, we use adaptive number of subsets per stage to allow effective and flexible feature encoding. In our STDA encoder, we split the input tensor $H \times W \times C$ into s subsets, each subset is denoted by $\boldsymbol{x}_i$ and has the same spatial size with C/s channels, where $i \in \{1, 2, ..., s\}$ and C is the number of channels. Each feature maps subset (except the first subset) is passed to 3×3 depth-wise convolution, denoted by d_i, and the output is denoted by $\boldsymbol{y}_i$. Also, the output of d_{i-1}, denoted by $\boldsymbol{y}_{i-1}$, is added to the feature subset $\boldsymbol{x}_i$, and then fed into d_i. The number of subsets s is adaptive based on the stage number t, where $t \in \{2, 3, 4\}$. We can write $\boldsymbol{y}_i$ as follows:

$$\boldsymbol{y}_i = \begin{cases} \boldsymbol{x}_i & i = 1; \\ d_i(\boldsymbol{x}_i) & i = 2, t = 2; \\ d_i(\boldsymbol{x}_i + \boldsymbol{y}_{i-1}) & 2 < i \leq s, t. \end{cases} \tag{2}$$

Each depth-wise operation d_i, as shown in SDTA encoder in Fig. 2, receives feature maps output from all previous splits $\{\boldsymbol{x}_j, j \leq i\}$.

As mentioned earlier, the overhead of the transformer self-attention layer is infeasible for vision tasks on edge-devices because it comes at the cost of higher MAdds and latency. To alleviate this issue and encode the global context efficiently, we use transposed query and key attention feature maps in our SDTA encoder [1]. This operation has a linear complexity by applying the dot-product operation of the MSA across channel dimensions instead of the spatial dimension, which allows us to compute cross-covariance across channels to generate attention feature maps that have implicit knowledge about the global representations. Given a normalized tensor $\boldsymbol{Y}$ of shape $H \times W \times C$, we compute query ($\boldsymbol{Q}$), key ($\boldsymbol{K}$), and value ($\boldsymbol{V}$) projections using

three linear layers, yielding $Q=W^QY$, $K=W^KY$, and $V=W^VY$, with dimensions $HW \times C$, where W^Q, W^K, and W^V are the projection weights for Q, K, and V respectively. Then, L2 norm is applied to Q and K before computing the cross-covariance attention as it stabilizes the training. Instead of applying the dot-product between Q and K^T along the spatial dimension i.e., $(HW \times C) \cdot (C \times HW)$, we apply the dot-product across the channel dimensions between Q^T and K i.e., $(C \times HW)$ $\cdot (HW \times C)$, producing $C \times C$ softmax scaled attention score matrix. To get the final attention maps, we multiply the scores by V and sum them up. The transposed attention operation can be expressed as follows:

$$\hat{X} = Attention(Q,\ K,\ V) + X, \tag{3}$$

$$s.t.,\ Attention(Q,\ K,\ V) = V \cdot \texttt{softmax}(Q^T \cdot K) \tag{4}$$

where X is the input and $\hat{X}$ is the output feature tensor. After that, two 1×1 pointwise convolution layers, LN and GELU activation are used to generate non-linear features. Table 1 shows the sequence of Conv. and STDA encoders with the corresponding input size at each layer with more design details about extra-extra small, extra-small and small models.

4   Experiments

In this section, we evaluate our EdgeNeXt model on ImageNet-1K classification, COCO object detection, and Pascal VOC segmentation benchmarks.

4.1   Dataset

We use ImageNet-1K [35] dataset in all classification experiments. The dataset provides approximately 1.28M training and 50K validation images for 1000 categories. Following the literature [17,29], we report top-1 accuracy on the validation set for all experiments. For object detection, we use COCO [22] dataset which provides approximately 118k training and 5k validation images respectively. For segmentation, we use Pascal VOC 2012 dataset [10] which provides almost 10k images with semantic segmentation masks. Following the standard practice as in [29], we use extra data and annotations from [22] and [13] as well.

4.2   Implementation Details

We train our EdgeNeXt models at an input resolution of 256×256 with an effective batch size of 4096. All the experiments are run for 300 epochs with AdamW [27] optimizer, and with a learning rate and weight decay of 6e-3 and 0.05 respectively. We use cosine learning rate schedule [26] with linear warmup for 20 epochs. The data augmentations used during training are Random Resized Crop (RRC), Horizontal Flip, and RandAugment [6], where RandAugment is only used for the EdgeNeXt-S model. We also use multi-scale sampler [29] during training. Further stochastic depth [18] with

Table 1: **EdgeNeXt Architectures.** Description of the models' layers with respect to output size, kernel size, and output channels, repeated n times, along with the models MAdds and parameters. The number of the output channels for small, extra-small, and extra-extra small models is chosen to match the number of parameters with the counterpart MobileViT model. We use adaptive kernel sizes in Conv. Encoder to reduce the model complexity and capture different levels of features. Also, we pad the output size of the last stage to be able to apply the 9×9 filter.

Layer	Output Size	n	Kernel	Output Channels		
				XXS	**XS**	**S**
Image	256×256	1	-	-	-	-
Stem	64×64	1	4×4	24	32	48
Conv. Encoder	64×64	3	3×3	24	32	48
Downsampling	32×32	1	2×2	48	64	96
Conv. Encoder	32×32	2	5×5	48	64	96
STDA Encoder	32×32	1	-	48	64	96
Downsampling	16×16	1	2×2	88	100	160
Conv. Encoder	16×16	8	7×7	88	100	160
STDA Encoder	16×16	1	-	88	100	160
Downsampling	8×8	1	2×2	168	192	304
Conv. Encoder	8×8	2	9×9	168	192	304
STDA Encoder	8×8	1	-	168	192	304
Global Average Pooling	1×1	1	-	-	-	-
Linear	1×1	1	-	1000	1000	1000
Model MAdds				0.3G	0.5G	1.3G
Model Prameters				1.3M	2.3M	5.6M

a rate of 0.1 is used for EdgeNeXt-S model only. We use EMA [32] with a momentum of 0.9995 during training. For inference, the images are resized to 292×292 followed by a center crop at 256×256 resolution. We also train and report the accuracy of our EdgeNeXt-S model at 224×224 resolution for a fair comparison with previous methods. The classification experiments are run on eight A100 GPUs with an average training time of almost 30 hours for the EdgeNeXt-S model.

For detection and segmentation tasks, we finetune EdgeNeXt following similar settings as in [29] and report mean average precision (mAP) at IOU of 0.50-0.95 and mean intersection over union (mIOU) respectively. The experiments are run on four A100 GPUs with an average training time of ∼36 and ∼7 hours for detection and segmentation respectively.

We also report the latency of our models on NVIDIA Jetson Nano[1] and NVIDIA A100 40GB GPU. For Jetson Nano, we convert all the models to TensorRT[2] engines

---

[1] https://developer.nvidia.com/embedded/jetson-nano-developer-kit
[2] https://github.com/NVIDIA/TensorRT

Table 2: Comparisons of our proposed EdgeNeXt model with state-of-the-art lightweight fully convolutional, transformer-based and hybrid models on ImageNet-1K classification task. Our model achieves a better trade-off between accuracy and compute (i.e., parameters and multiplication-addition (MAdds)).

Frameworks	Models	Date	Input	Params↓	MAdds↓	Top1↑
ConvNets	MobileNetV2	CVPR2018	224^2	6.9M	585M	74.7
	ShuffleNetV2	ECCV2018	224^2	5.5M	597M	74.5
	MobileNetV3	ICCV2019	224^2	5.4M	219M	75.2
ViTs	T2T-ViT	ICCV2021	224^2	6.9M	1.80G	76.5
	DeiT-T	ICML2021	224^2	5.7M	1.25G	72.2
Hybrid	MobileFormer	CoRR2021	224^2	9.4M	214M	76.7
	ViT-C	NeurIPS2021	224^2	4.6M	1.10G	75.3
	CoaT-Lite-T	ICCV2021	224^2	5.7M	1.60G	77.5
	MobileViT-S	ICLR2022	256^2	5.6M	2.01G	78.4
	EdgeNeXt-S	Ours	224^2	5.6M	965M	78.8
	EdgeNeXt-S	Ours	256^2	5.6M	1.30G	79.4

and perform inference in FP16 mode using a batch size of 1. For A100, similar to [25], we use PyTorch v1.8.1 with a batch size of 256 to measure the latency.

4.3   Image Classification

Table 2 compares our proposed EdgeNeXt model with previous state-of-the-art fully convolutional (ConvNets), transformer-based (ViTs) and hybrid models. Overall, our model demonstrates better accuracy versus compute (parameters and MAdds) trade-off compared to all three categories of methods (see Fig. 1).

Comparison with ConvNets. EdgeNeXt surpasses ligh-weight ConvNets by a formidable margin in terms of top-1 accuracy with similar parameters (Table 2). Normally, Con-

Table 3: Comparison of different variants of EdgeNeXt with the counterpart models of Mobile-ViT. The last two columns list the latency in ms and μs on NVIDIA Jetson Nano and A100 devices, respectively. Our EdgeNext models provide higher accuracy with lower latency for each model size, indicating the flexibility of our design to scale down to as few as 1.3M parameters.

Model	Date	Input	Params↓	MAdds↓	Top1↑	Nano↓	A100↓
MobileViT-XXS			1.3M	364M	69.0	$21.0\,ms$	$216\,\mu s$
MobileViT-XS	ICLR2022	256^2	2.3M	886M	74.8	$35.1\,ms$	$423\,\mu s$
MobileViT-S			5.6M	2.01G	78.4	$53.0\,ms$	$559\,\mu s$
EdgeNeXt-XXS			1.3M	261M	71.2	$19.3\,ms$	$142\,\mu s$
EdgeNeXt-XS	Ours	256^2	2.3M	538M	75.0	$31.6\,ms$	$227\,\mu s$
EdgeNeXt-S			5.6M	1.30G	79.4	$48.8\,ms$	$332\,\mu s$

vNets have less MAdds compared to transformer and hybrid models because of no attention computation, however, they lack the global receptive field. For instance, EdgeNeXt-S has higher MAdds compared to MobileNetV2 [36], but it obtains 4.1% gain in top-1 accuracy with less number of parameters. Also, our EdgeNeXt-S outperforms ShuffleNetV2 [28] and MobileNetV3 [16] by 4.3% and 3.6% respectively, with comparable number of parameters.

Comparison with ViTs. Our EdgeNeXt outperforms recent ViT variants on ImageNet1K dataset with fewer parameters and MAdds. For example, EdgeNeXt-S obtains 78.8% top-1 accuracy, surpassing T2T-ViT [44] and DeiT-T [40] by 2.3% and 6.6% absolute margins respectively.

Comparison with Hybrid Models. The proposed EdgeNeXt outperforms MobileFormer [4], ViT-C [42], CoaT-Lite-T [7] with less number of parameters and fewer MAdds (Table 2). For a fair comparison with MobileViT [29], we train our model at an input resolution of 256×256 and show consistent gains for different models sizes (i.e., S, XS, and XXS) with fewer MAdds and faster inference on the edge devices (Table 3). For instance, our EdgeNeXt-XXS model achieves 71.2% top-1 accuracy with only 1.3M parameters, surpassing the corresponding MobileViT version by 2.2%. Finally, our EdgeNeXt-S model attains 79.4% accuracy on ImageNet with only 5.6M parameters, a margin of 1.0% as compared to the corresponding MobileViT-S model. This demonstrates the effectiveness and the generalization of our design.

Further, we also train our EdgeNeXt-S model using knowledge distillation following [34] and achieves 81.1% top-1 ImageNet accuracy.

4.4 ImageNet-21K Pretraining

To further explore the capacity of EdgeNeXt, we designed EdgeNeXt-B model with 18.5M parameters and 3.8MAdds and pretrain it on a subset of ImageNet-21K [35] dataset followed by finetuning on standard ImageNet-1K dataset. ImageNet-21K (winter'21 release) contains around 13M images and 19K classes. We follow [33] to preprocess the pretraining data by removing classes with fewer examples and split it into training and validation sets containing around 11M and 522K images respectively over 10,450 classes. We refer this dataset as ImageNet-21K-P. We strictly follow the training recipes of [25] for ImageNet-21K-P pretaining. Further, we initialize the ImageNet-21K-P training with ImageNet-1K pretrained model for faster convergence. Finally, we finetune ImageNet-21K model on ImageNet-1K for 30 epochs with a learning rate of $7.5e^{-5}$ and an effective batch size of 512. The results are summarized in Table 4.

4.5 Inference on Edge Devices

We compute the inference time of our EdgeNeXt models on the NVIDIA Jetson Nano edge device and compare it with the state-of-the-art MobileViT [29] model (Table 3). All the models are converted to TensorRT engines and inference is performed in FP16 mode. Our model attains low latency on the edge device with similar parameters, fewer MAdds, and higher top-1 accuracy. Table 3 also lists the inference time on A100 GPU for both MobileViT and EdgeNeXt models. It can be observed that our EdgeNeXt-XXS model is ∼34% faster than the MobileViT-XSS model on A100 as compared to

Table 4: Large-scale ImageNet-21K-P pretraining of EdgeNeXt-B model. Our model achieves better accuracy vs compute trade-off compared to SOTA ConvNeXt [25] and MobileViT-V2 [30].

Model	Pretraining	Input	Params↓	MAdds↓	Top1↑
ConvNeXt-T	None	224^2	28.6M	4.5G	82.1
ConvNeXt-T	ImageNet-21K	224^2	28.6M	4.5G	82.9
MobileViT-V2	None	256^2	18.5M	7.5G	81.2
MobileViT-V2	ImageNet-21K-P	256^2	18.5M	7.5G	82.4
EdgeNeXt-B	None	256^2	18.5M	3.8G	82.5
EdgeNeXt-B	ImageNet-21K-P	256^2	18.5M	3.8G	83.3

only ~8% faster on Jetson Nano, indicating that EdgeNeXt better utilizes the advanced hardware as compared to MobileViT.

4.6 Object Detection

We use EdgeNeXt as a backbone in SSDLite and finetune the model on COCO 2017 dataset [22] at an input resolution of 320×320. The difference between SSD [23] and SSDLite is that the standard convolutions are replaced with separable convolutions in the SSD head. The results are reported in Table 5. EdgeNeXt consistently out-performs MobileNet backbones and gives competitive performance compared to Mo-

Model	Params↓	MAdds↓	mAP↑
MobileNetV1	5.1M	1.3G	22.2
MobileNetV2	4.3M	800M	22.1
MobileNetV3	5.0M	620M	22.0
MobileViT-S	5.7M	3.4G	27.7
EdgeNeXt-S (ours)	6.2M	2.1G	27.9

Table 5: Comparisons with SOTA on COCO object detection. EdgeNeXt improves over previous approaches.

bileVit backbone. With less number of MAdds and a comparable number of parameters, EdgeNeXt achieves the highest 27.9 box AP, ~38% fewer MAdds than MobileViT.

4.7 Semantic Segmentation

We use EdgeNeXt as backbone in DeepLabv3 [3] and finetune the model on Pascal VOC [10] dataset at an input resolution of 512×512. DeepLabv3 uses dilated convolution in cascade design along with spatial pyramid pooling to encode multi-scale features which are useful in encoding objects at multiple

Table 6: Comparisons with SOTA on VOC semantic segmentation. Our model provides reasonable gains.

Model	Params↓	MAdds↓	mIOU↑
MobileNetV1	11.1M	14.2G	75.3
MobileNetV2	4.5M	5.8G	75.7
MobileViT-S	5.7M	13.7G	79.1
EdgeNeXt-S (ours)	6.5M	8.7G	80.2

scales. Our model obtains 80.2 mIOU on the validation dataset, providing a 1.1 points gain over MobileViT with ~36% fewer MAdds.

5   Ablations

In this section, we ablate different design choices in our proposed EdgeNeXt model.
SDTA encoder and adaptive kernel sizes. Table 7 illustrates the importance of SDTA encoders and adaptive kernel sizes in our proposed architecture. Replacing SDTA encoders with convolution encoders degrades the accuracy by 1.1%, indicating the usefulness of SDTA encoders in our design. When we fix kernel size to 7 in all four stages of the network, it further reduces the accuracy by 0.4%. Overall, our proposed design provides an optimal speed-accuracy trade-off.

We also ablate the contributions of SDTA components (e.g., adaptive branching and positional encoding) in Table 7. Removing adaptive branching and positional encoding slightly decreases the accuracy.

	Model	Top1↑	Latency↓
Base	EdgeNeXt-S	79.4	332 μs
Different Components	w/o SDTA Encoders	78.3	265 μs
	+ w/o Adaptive Kernels	77.9	301 μs
SDTA Components	w/o Adaptive Branching	79.3	332 μs
	+ w/o PE	79.2	301 μs

Table 7: Ablation on different components of EdgeNeXt and SDTA encoder. The results show the benefits of SDTA encoders and adaptive kernels in our design. Further, adaptive branching and positional encoding (PE) are also required in SDTA module.

Hybrid design. Table 8 ablates the different hybrid design choices for our EdgeNeXt model. Motivated from MetaFormer [43], we replace all convolutional modules in the last two stages with SDTA encoders. The results show superior performance when all blocks in the last two stages are SDTA blocks, but it increases the latency (row-2 vs 3). Our hybrid design where we propose to use an SDTA module as the last block in the last three stages provides an optimal speed-accuracy trade-off.

Model Configuration	Top1↑	Latency↓
1: Conv=[3, 3, 9, 0], SDTA=[0, 0, 0, 3]	79.3	303 μs
2: Conv=[3, 3, 0, 0], SDTA=[0, 0, 9, 3]	79.7	393 μs
3: Conv=[3, 2, 8, 2], SDTA=[0, 1, 1, 1]	79.4	332 μs

Table 8: Ablation on the hybrid architecture. Using one SDTA encoder as the last block in the last three stages provides an optimal accuracy-latency trade-off.

Table 9 provides an ablation of the importance of using SDTA encoders at different stages of the network. It is noticeable that progressively adding an SDTA encoder as the last block of the last three stages improves the accuracy with some loss in inference latency. However, in row 4, we obtain the best trade-off between accuracy and speed where the SDTA encoder is added as the last block in the last three stages of the network. Further, we notice that adding a global SDTA encoder to the first stage of the network is not helpful where the features are not much mature.

We also provide an ablation on using the SDTA module at the start of each stage versus at the end. Table 10 shows that using the global SDTA encoder at the end of each stage is more beneficial. This observation is consistent with the recent work [21].
Activation and normalization. EdgeNeXt uses GELU activation and layer normalization throughout the network. We found that the current PyTorch implementations of GELU and layer normalization are not optimal for high speed inference. To this end, we replace GELU with Hard-Swish and layer-norm with batch-norm and retrain our mod-

Model Configuration	Top1↑	Latency↓
1: Conv=[3, 3, 9, 3], SDTA=[0, 0, 0, 0]	78.3	265 μs
2: Conv=[3, 3, 9, 2], SDTA=[0, 0, 0, 1]	78.6	290 μs
3: Conv=[3, 3, 8, 2], SDTA=[0, 0, 1, 1]	79.1	310 μs
4: Conv=[3, 2, 8, 2], SDTA=[0, 1, 1, 1]	79.4	332 μs
5: Conv=[2, 2, 8, 2], SDTA=[1, 1, 1, 1]	79.2	387 μs

Table 9: Ablation on using SDTA encoder at different stages of the network. Including SDTA encoders in the last three stages improves performance, whereas a global SDTA encoder is not helpful in the first stage.

SDTA Configuration	Top1↑	Latency↓
Start of Stage (SDTA=[0, 1, 1, 1])	79.0	332 μs
End of Stage (SDTA=[0, 1, 1, 1])	79.4	332 μs

Table 10: Ablation on using SDTA at the start and end of each stage in EdgeNeXt. The results show that it is generally beneficial to use SDTA at the end of each stage.

els. Fig. 3 indicates that it reduces the accuracy slightly, however, reduces the latency by a large margin.
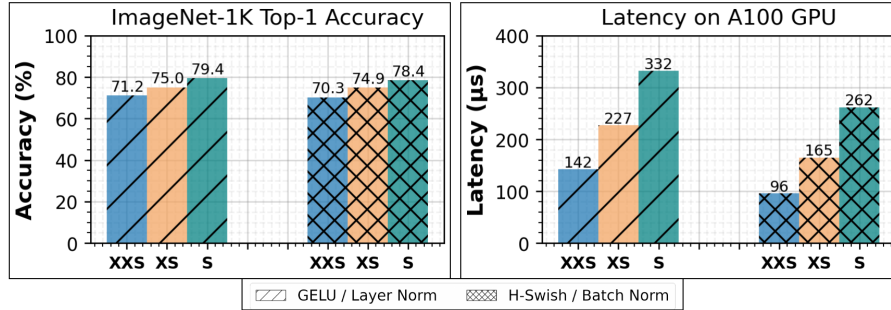

Fig. 3: Ablation on the effect of using different activation functions and normalization layers on accuracy and latency of our network variants. Using Hard Swish activation and batch normalization instead of GELU and layer normalization significantly improves the latency at the cost of some loss in accuracy.

6 Qualitative Results

Figs. 4 and 5 shows the qualitative results of EdgeNeXt detection and segmentation models respectively. Our model can detect and segment objects in various views.

7 Conclusion

The success of the transformer models comes with a higher computational overhead compared to CNNs. Self-attention operation is the major contributor to this overhead, which makes vision transformers slow on the edge devices compared to CNN-based mobile architectures. In this paper, we introduce a hybrid design consisting of convolution and efficient self-attention based encoders to jointly model local and global information effectively, while being efficient in terms of both parameters and MAdds on vision tasks with superior performance compared to state-of-the-art methods. Our exper-
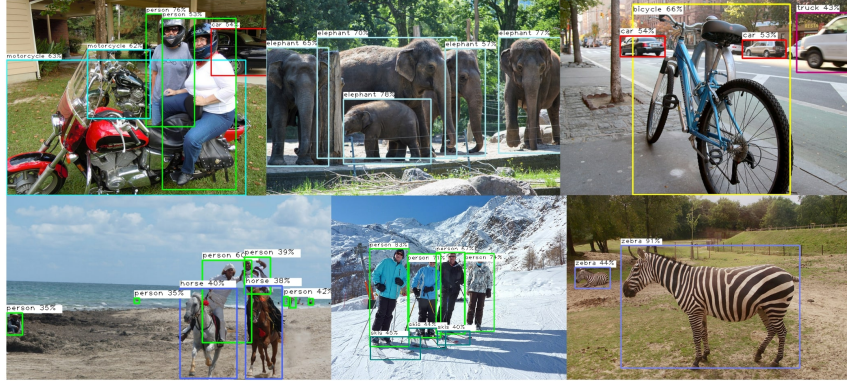
Fig. 4: Qualitative results of our EdgeNeXt detection model on COCO validation dataset. The model is trained on COCO dataset with 80 detection classes. Our model can effectively localize and classify objects in diverse scenes.
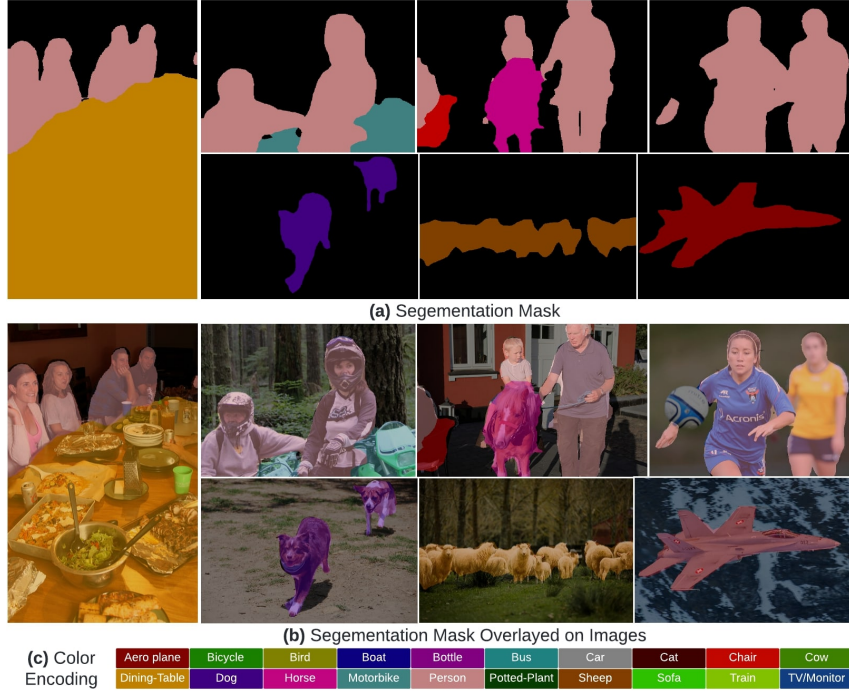

(a) Segementation Mask

(b) Segmentation Mask Overlayed on Images

(c) Color	Aero plane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow
Encoding	Dining-Table	Dog	Horse	Motorbike	Person	Potted-Plant	Sheep	Sofa	Train	TV/Monitor

Fig. 5: Qualitative results of our EdgeNeXt segmentation model on unseen COCO validation dataset. The model is trained on Pascal VOC dataset with 20 segmentation classes. **(a)** shows the predicted semantic segmentation mask where 'black 'color represents the background pixels. **(b)** displays the predicted masks on top of original images. **(c)** represents the color encodings for all Pascal VOC classes for the displayed segmentation masks. Our model provides high-quality segmentation masks on unseen COCO images.

imental results show promising performance for different variants of EdgeNeXt, which demonstrates the effectiveness and the generalization ability of the proposed model.

References

1. Ali, A., Touvron, H., Caron, M., Bojanowski, P., Douze, M., Joulin, A., Laptev, I., Neverova, N., Synnaeve, G., Verbeek, J., et al.: Xcit: Cross-covariance image transformers. Advances in Neural Information Processing Systems (2021) 6

2. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016) 6

3. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017) 11

4. Chen, Y., Dai, X., Chen, D., Liu, M., Dong, X., Yuan, L., Liu, Z.: Mobile-former: Bridging mobilenet and transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022) 3, 10

5. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2017) 3

6. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.: Randaugment: Practical automated data augmentation with a reduced search space. In: Advances in Neural Information Processing Systems (2020) 7

7. Dai, Z., Liu, H., Le, Q., Tan, M.: Coatnet: Marrying convolution and attention for all data sizes. Advances in Neural Information Processing Systems (2021) 10

8. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020) 2, 3

9. d'Ascoli, S., Touvron, H., Leavitt, M.L., Morcos, A.S., Biroli, G., Sagun, L.: Convit: Improving vision transformers with soft convolutional inductive biases. In: International Conference on Machine Learning (2021) 3

10. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The Pascal Visual Object Classes (VOC) Challenge. International Journal of Computer Vision **88**(2), 303–338 (2010) 7, 11

11. Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., Feichtenhofer, C.: Multiscale vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021) 3

12. Gao, S.H., Cheng, M.M., Zhao, K., Zhang, X.Y., Yang, M.H., Torr, P.: Res2net: A new multiscale backbone architecture. IEEE Transactions on Pattern Analysis and Machine Intelligence **43**(2), 652–662 (2019) 6

13. Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2011) 7

14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2016) 4

15. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016) 6

16. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2019) 2, 3, 10

17. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR **abs/1704.04861** (2017) 3, 7

18. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: The European Conference on Computer Vision (2016) 7

19. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016) 2, 3
20. Khan, S., Naseer, M., Hayat, M., Zamir, S.W., Khan, F.S., Shah, M.: Transformers in vision: A survey. ACM Computing Surveys (CSUR) (2021) 1
21. Li, Y., Wu, C.Y., Fan, H., Mangalam, K., Xiong, B., Malik, J., Feichtenhofer, C.: Improved multiscale vision transformers for classification and detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022) 12
22. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: The European Conference on Computer Vision (2014) 7, 11
23. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: The European Conference on Computer Vision (2016) 11
24. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021) 2, 3
25. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022) 4, 5, 9, 10, 11
26. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. In: International Conference on Learning Representations (2017) 7
27. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2019) 7
28. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: The European Conference on Computer Vision (2018) 2, 10
29. Mehta, S., Rastegari, M.: Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. In: International Conference on Learning Representations (2022) 2, 3, 4, 7, 8, 10
30. Mehta, S., Rastegari, M.: Separable self-attention for mobile vision transformers. arXiv preprint arXiv:2206.02680 (2022) 11
31. Mehta, S., Rastegari, M., Shapiro, L., Hajishirzi, H.: Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019) 3
32. Polyak, B.T., Juditsky, A.B.: Acceleration of stochastic approximation by averaging. SIAM journal on control and optimization (1992) 8
33. Ridnik, T., Ben-Baruch, E., Noy, A., Zelnik-Manor, L.: Imagenet-21k pretraining for the masses. arXiv preprint arXiv:2104.10972 (2021) 10
34. Ridnik, T., Lawen, H., Ben-Baruch, E., Noy, A.: Solving imagenet: a unified scheme for training any backbone to top results. arXiv preprint arXiv:2204.03475 (2022) 10
35. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision **115**(3), 211–252 (2015) 3, 7, 10
36. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018) 2, 3, 10
37. Schmidhuber, J.: Deep learning in neural networks: An overview. Neural networks **61**, 85–117 (2015) 1
38. Srinivas, A., Lin, T.Y., Parmar, N., Shlens, J., Abbeel, P., Vaswani, A.: Bottleneck transformers for visual recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021) 3

39. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019) 2, 3

40. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning (2021) 3, 10

41. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (2017) 3

42. Xiao, T., Dollar, P., Singh, M., Mintun, E., Darrell, T., Girshick, R.: Early convolutions help transformers see better. In: Advances in Neural Information Processing Systems (2021) 3, 10

43. Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., Yan, S.: Metaformer is actually what you need for vision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021) 12

44. Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z.H., Tay, F.E., Feng, J., Yan, S.: Tokens-to-token vit: Training vision transformers from scratch on imagenet. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021) 10

45. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: The European Conference on Computer Vision (2014) 4

46. Zhang, H., Hu, W., Wang, X.: Edgeformer: Improving light-weight convnets by learning from vision transformers. arXiv preprint arXiv:2203.03952 (2022) 2

47. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018) 3

48. Zhou, J., Wang, P., Wang, F., Liu, Q., Li, H., Jin, R.: Elsa: Enhanced local self-attention for vision transformer. arXiv preprint arXiv:2112.12786 (2021) 3