SwiftFormer: Efficient Additive Attention for Transformer-based Real-time Mobile Vision Applications

Abdelrahman Shaker[1*] Muhammad Maaz[1] Hanoona Rasheed[1] Salman Khan[1]
Ming-Hsuan Yang[2,3,4] Fahad Shahbaz Khan[1,5]
[1]Mohamed bin Zayed University of AI [2]University of California, Merced
[3]Yonsei University [4]Google Research [5]Linköping University

Abstract

Self-attention has become a defacto choice for capturing global context in various vision applications. However, its quadratic computational complexity with respect to image resolution limits its use in real-time applications, especially for deployment on resource-constrained mobile devices. Although hybrid approaches have been proposed to combine the advantages of convolutions and self-attention for a better speed-accuracy trade-off, the expensive matrix multiplication operations in self-attention remain a bottleneck. In this work, we introduce a novel efficient additive attention mechanism that effectively replaces the quadratic matrix multiplication operations with linear element-wise multiplications. Our design shows that the key-value interaction can be replaced with a linear layer without sacrificing any accuracy. Unlike previous state-of-the-art methods, our efficient formulation of self-attention enables its usage at all stages of the network. Using our proposed efficient additive attention, we build a series of models called "SwiftFormer" which achieves state-of-the-art performance in terms of both accuracy and mobile inference speed. Our small variant achieves 78.5% top-1 ImageNet-1K accuracy with only 0.8 ms latency on iPhone 14, which is more accurate and 2× faster compared to MobileViT-v2. Our code and models: `https://tinyurl.com/5ft8v46w`

1. Introduction

In recent years, transformer models have shown remarkable success in various vision applications such as classification [8, 44, 24, 23, 9], detection [2, 59, 33, 56, 28], and segmentation [4, 40]. However, deploying these models on resource-constrained mobile devices for real-time applications remains challenging due to their inherently complex nature [20, 29]. Specifically, vision transformers (ViTs) rely on global self-attention, which has a quadratic complexity with respect to the input image resolution, making it impractical for deployment on low-powered mobile devices [31]. As a result, convolutional neural networks (CNNs) are still the preferred choice for real-time deployment on mobile devices, primarily because the convolution operation is computationally efficient [39, 15]. However, a major limitation of CNNs is their reliance on local connections and stationary weights, which can limit their ability to adapt to variable input resolutions and capture long-range dependencies in the data. Therefore, developing more efficient and flexible models that combine the strengths of both CNNs and transformers is critical, particularly for mobile devices with limited computational resources.

To achieve this goal, several hybrid approaches have

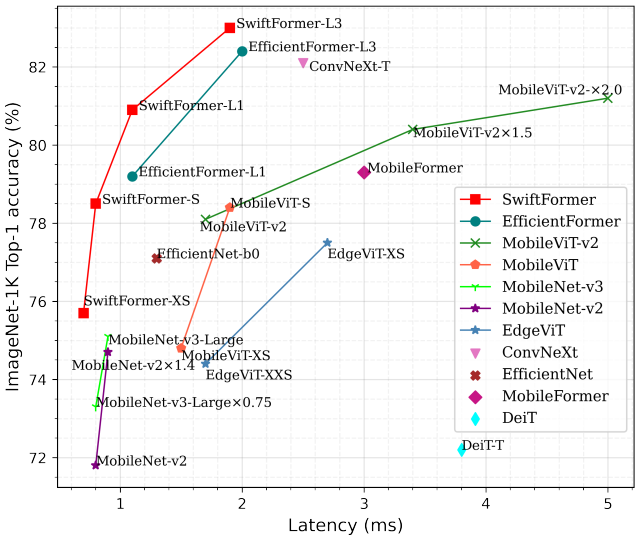

Figure 1: **Latency vs Accuracy Comparison**. Compared to the recent EfficientFormer-L1 [20], our SwiftFormer-L1 achieves an absolute gain of 1.7% in terms of top-1 accuracy with the same latency and without requiring any neural architecture search.

*Corresponding author: abdelrahman.youssief@mbzuai.ac.ae

been proposed that use lightweight CNN modules in the high-resolution early stages and self-attention in the low-resolution later stages [55, 29, 20]. This approach effectively increases the receptive field of the network and strives to achieve a trade-off between speed and accuracy. Furthermore, different efficient variants of computing self-attention have been proposed to reduce the model complexity. These include computing attention across feature dimensions to implicitly model the global context [29], computing attention within local windows [24], pooling spatial features before applying self-attention [9], and sparsely attending to a fixed number of tokens [34], to name a few.

Although these approaches effectively reduce network complexity, they still involve inefficient matrix multiplication operations that significantly impact latency on mobile devices. To address this issue, Mehta et al. [31] propose a separable self-attention mechanism that replaces matrix multiplication operations to element-wise multiplications. This is achieved by projecting queries to context scores, followed by element-wise multiplication with keys to calculate context vectors for encoding global context.

In this work, we propose efficient additive attention, which eliminates the need for expensive matrix multiplication operations in computing self-attention. Additionally, we propose to compute the global context using only the query-key interactions followed by a linear transformation, without requiring explicit key-value interactions. This significantly reduces the computational complexity and enables us to use the proposed attention block in all stages of the network. Our contributions are as follows:

- We introduce *efficient additive attention*, a new approach for computing self-attention in vision backbones that eliminates the need for expensive matrix multiplication operations, significantly reducing the computational complexity of the model.

- Unlike previous methods, our proposed efficient attention design can be used at all stages of the network, enabling more effective contextual information capture and achieving superior speed-accuracy trade-off.

- We build a series of efficient generic classification models called "SwiftFormer", which utilize our proposed *efficient additive attention*. Our *small* model achieves 78.5% top-1 ImageNet-1K [7] accuracy while running at only 0.8 ms latency on iPhone 14. Moreover, our large model achieves 83.0% accuracy with a latency of only 1.9 ms. Our model achieves state-of-the-art performance, outperforming recent MobileViT-v2 [31] and EfficientFormer [20] by obtaining a better trade-off between accuracy and latency (see Fig. 1).

2. Related Work

Efficient CNNs: Designing efficient CNNs for mobile vision applications has received much attention in recent years. MobileNet architectures [16, 39, 15] propose depth-wise separable convolutions as well as efficient inverted residual blocks for improved performance on various vision tasks. Other methods aim to improve the efficiency by leveraging depth-wise dilated convolutions [32], channel shuffling and pointwise group convolutions [57, 27], network pruning [11, 50], low bit-width [1, 17], and neural architecture search [41, 15]. CNN-based methods are well-performing, efficient, and fast to train and run on edge devices, resulting in widespread usage in the industry. However, they are spatially local and lack global interaction between the features, which deeply affects their performance.
Efficient Transformers: ViTs [8] have been widely used in numerous vision tasks, and significant advances have been made in terms of data efficiency [22, 44], transformer architecture [30, 20, 3, 54], and token mechanisms [43, 52]. Reducing the number of visual tokens is a major modification in the transformer architecture for efficient deployment. Instead of using a fixed feature representation through the whole architecture, some methods employ a hierarchical design where the resolution is gradually decreased through the stages, including down-sampling techniques [9, 40, 38, 14] and pyramidal structures [47, 49]. Recently, a few methods [37, 10, 34] propose token sparsification techniques to encode only a subset of the most informative tokens.

Numerous approaches have recently been proposed to reduce the quadratic complexity of self-attention, the computational bottleneck in transformer-based architectures, by computing its approximated variants [34, 29, 31, 46, 19, 5, 45]. EdgeViT [34] uses a global sparse attention module attending only to a few tokens to improve the efficiency, while [48] down-samples the key and value vectors that lead to a better efficiency-accuracy trade-off. GCViT [12] uses a local-global attention mechanism by employing the interactions between a global query and local key and value to capture the contextual information. This approach enables the model to encode both local and global representations efficiently. EdgeNeXt [29] adopts transposed self-attention operation to compute the attention maps across the channel dimension instead of the spatial dimension, followed by token mixing, to have a linear complexity with respect to the number of tokens. Reformer [19] replaces the dot-product attention with a locality-sensitive hashing to group the tokens and reduced the complexity from $O(n^2)$ to $O(n \log n)$. However, this design is only efficient on longer sequences, which is typically not the case for ViTs. LinFormer [46] is a low-rank matrix factorization method that approximates the self-attention matrix with a low-rank matrix, reducing the complexity from $O(n^2)$ to $O(n)$. Although matrix factorization methods theoretically reduce the complexity of

self-attention, they use expensive projections for computing attention, which may not reflect the reduction in FLOPs and parameters into actual speed on mobile platforms.

Although these methods show promise and have reduced the complexity of self-attention theoretically, they are inadequate for reducing the inference speed for mobile deployment. Since the complexity of the multi-headed self-attention (MHSA) is higher in the earlier stage compared to the last stages, EfficientFormer [20] incorporates MHSA in the last stage only to learn contextual information from the high-level features without increasing the inference speed significantly. Recently, MobileViT-v2 [31] proposes separable self-attention that uses element-wise operations instead of the dot-product to compute the attention maps with linear complexity. Different from the existing approaches, we propose a consistent hybrid design with an *efficient additive attention* mechanism to model the contextual information with linear complexity. Instead of capturing the pairwise interactions between keys, queries, and values using the dot-product, we use element-wise operations with learnable attention weights to model the interactions between query and keys only, leading to better inference speed.

3. Method

Motivation: To motivate our method, we first distinguish three desirable characteristics to be considered when designing an efficient yet accurate approach for resource constraint mobile devices.

Efficient Global Context Modeling: As discussed earlier, most existing approaches either employ the standard MHSA or an approximated variant to learn the global context. However, they struggle to operate as fast as MobileNets on resource-constrained devices. This is likely due to the computation-intensive multiplicative operations during attention computation or reliance on advanced reshaping and indexing operations in these approaches. For instance, the recent MobileViT-v2 [31] is 2$\times$ slower than MobileNet-v2 [39]. Instead of using matrix multiplications, we argue that encoding the global context using an efficient additive attention design can reduce the operations with respect to the number of tokens. This is expected to help operate at comparable speed and model size, while achieving superior accuracy compared to MobileNets.

Rethinking key-value interactions: Other than multiplicative operations during attention computation, additive attention has been recently explored in the NLP domain [51]. However, in the standard form, it performs three-step processing to model query, key, and value interactions. Each step feeds into the subsequent one, thereby requiring sequential processing. Here, we rethink the additive attention for vision tasks by alleviating the need to compute explicit interactions between key-value. We argue that eliminating key-value interactions and replacing them with a simple linear transformation empirically encodes better contextual representation. Our design encodes only global queries and key interactions to learn the global contextual information, followed by a linear transformation to compute global context-aware attention weights.

Consistent Hybrid Design: Most existing works employ MHSA or the approximated variant in the last stages, while avoiding its usage in the earlier stages. This is because the computational complexity of MHSA grows quadratically with the length of the tokens, making it impractical to incorporate during the initial stages. This constraint adds to the design complexity and requires a careful selection of stages, where MHSA can be applied. In contrast, our proposed SwiftFormer module has linear complexity with respect to the token length and can be incorporated in all stages to learn consistent global context at each scale. This consistency improves the model performance and makes it more generalizable and scalable for high-resolution images.

3.1. Overview of Attention Modules

Vision transformer models are built upon the self-attention (see Fig. 2 (a)), which can effectively model the interactions between the input tokens. Specifically, the self-attention has $\mathbf{x}$ as an input, where $\mathbf{x} \in \mathbb{R}^{n \times d}$, comprising n tokens with d-dimensional embedding vector. The input $\mathbf{x}$ is projected to query ($\mathbf{Q}$), key ($\mathbf{K}$), and value ($\mathbf{V}$) using three matrices, $\mathbf{W_Q}$, $\mathbf{W_K}$, and $\mathbf{W_V}$. Each self-attention layer comprises h heads, which allows the model to attend to different views of the input. The self-attention can be described as:

$$\hat{\mathbf{x}} = \texttt{Softmax}\Big(\frac{\mathbf{Q} \cdot \mathbf{K}^\top}{\sqrt{d}}\Big) \cdot \mathbf{V}. \tag{1}$$

The attention scores between each pair of tokens in $\mathbf{Q}$ and $\mathbf{K}$ are computed using the dot-product operation. Next, these scores are normalized followed by Softmax to weigh the interactions between the tokens. Finally, the weighted interactions are multiplied by $\mathbf{V}$ using the dot-product operation to produce the final weighted output. Overall, the complexity of the self-attention is $O(n^2 \cdot d)$, where n is the number of tokens and d is the hidden dimension. The computational and memory demands of $\mathbf{Q} \cdot \mathbf{K}^\top$ increase quadratically as the number of tokens grows, leading to slow inference speed and high memory usage, making it impractical to run in real-time for long sequences.

To alleviate this issue, [29] proposes the transpose self-attention (see Fig. 2 (b)) to reduce the complexity from quadratic to linear with respect to the number of tokens. Here, the dot-product operation is applied across the channel dimension instead of the spatial dimension. This allows the model to learn feature maps with implicit contextual representation. The attention can be described as:

$$\hat{\mathbf{x}} = \mathbf{V} \cdot \texttt{Softmax}\Big(\frac{\mathbf{Q}^\top \cdot \mathbf{K}}{\sqrt{d}}\Big). \tag{2}$$

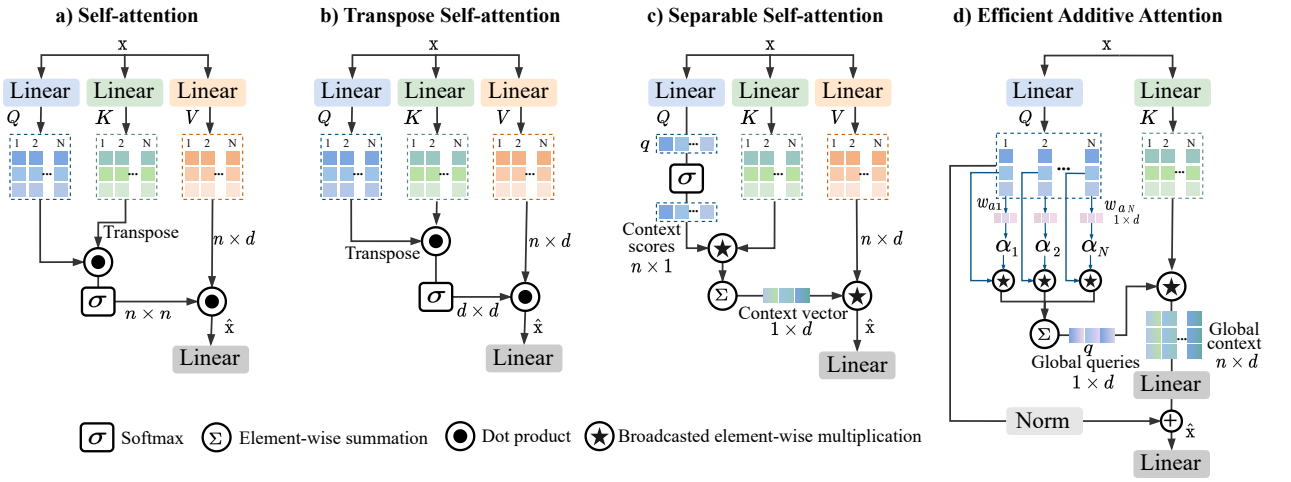

Figure 2: **Comparison with different self-attention modules**. **(a)** is a typical self-attention used in ViTs [8]. **(b)** is the transpose self-attention used in EdgeNeXt [29], where the self-attention operation is applied across channel feature dimensions (d×d) instead of the spatial dimension (n×n). **(c)** is the separable self-attention of MobileViT-v2 [31], it uses element-wise operations to compute the context vector from the interactions of **Q** and **K** matrices. Then, the context vector is multiplied by **V** matrix to produce the final output. **(d)** Our proposed efficient additive self-attention. Here, the query matrix is multiplied by learnable weights and pooled to produce global queries. Then, the matrix **K** is element-wise multiplied by the broadcasted global queries, resulting the global context representation.

The transpose self-attention has a computational complexity of $O(n \cdot d^2)$. While this complexity scales linearly with the number of tokens n, it remains quadratic with respect to the feature dimension d. Further, the dot-product operation is still utilized between the query and key matrices.

The separable self-attention mechanism (see Fig. 2 (c)) aims to address the bottleneck of the standard self-attention. Here, the interactions between the queries (**Q**), keys (**K**), and values (**V**) are encoded using element-wise operations. First, the query matrix **Q** is projected to produce a vector $\mathbf{q}$ of dimensions $n \times 1$, and then fed into Softmax to generate the context scores, which captures the importance of each query element. Then, the context scores are multiplied by the key matrix **K** and pooled to compute a context vector, which encodes the contextual information. Finally, the context vector is multiplied element-wise with the value matrix **V** to propagate the contextual information and produce the final output $\hat{\mathbf{x}}$. It can be summarized as:

$$\hat{\mathbf{x}} = \mathbf{V} * \sum \mathbf{K} * \texttt{Softmax}(\mathbf{q}). \tag{3}$$

Here, $*$ denotes the element-wise multiplication operation.

3.2. Efficient Additive Attention

The typical additive attention mechanism in NLP captures the global context by utilizing pairwise interactions between the tokens via element-wise multiplications instead of using dot-product operation. It encodes the relevance scores for the contextual information of the input sequence based on the interactions of the three attention components (**Q**, **K**, **V**). In contrast, we show that key-value interactions can be removed without sacrificing the performance and only focusing on effectively encoding query-key interactions by incorporating a linear projection layer is sufficient to learn the relationship between the tokens (see Fig. 2 (d)). This approach, named efficient additive attention, has a faster inference speed and produces more robust contextual representations as demonstrated by our performance on image classification, object detection, and segmentation tasks (Sec. 4). Specifically, the input embedding matrix $\mathbf{x}$ is transformed into query (**Q**) and key (**K**) using two matrices $\mathbf{W_q}$, $\mathbf{W_k}$, where $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{n \times d}$, $\mathbf{W_q}, \mathbf{W_k} \in \mathbb{R}^{d \times d}$, n is the token length and d is the dimensions of the embedding vector. Next, the query matrix **Q** is multiplied by learnable parameter vector $\mathbf{w}_a \in \mathbb{R}^d$ to learn the attention weights of the query, producing global attention query vector $\alpha \in \mathbb{R}^n$ as:

$$\alpha = \mathbf{Q} \cdot \mathbf{w}_a / \sqrt{d} \tag{4}$$

Then, the query matrix is pooled based on the learned attention weights, resulting in a single global query vector $\mathbf{q} \in \mathbb{R}^d$ as follows:

$$\mathbf{q} = \sum_{i=1}^{n} \alpha_i * \mathbf{Q_i}. \tag{5}$$

Next, the interactions between the global query vector $\mathbf{q} \in \mathbb{R}^d$ and key matrix $\mathbf{K} \in \mathbb{R}^{n \times d}$ are encoded using

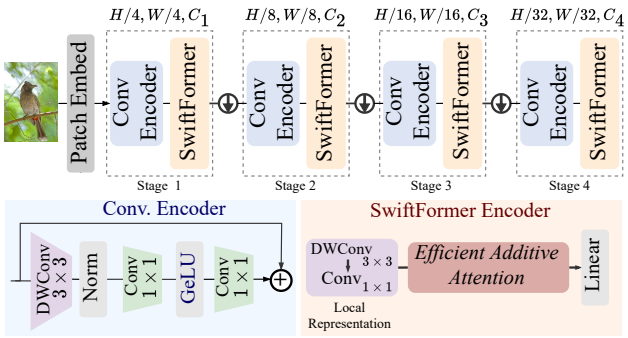

Figure 3: **Top Row:** Overview of our proposed architecture. The input image is fed into the patch embedding layer, followed by hierarchical stages at four different scales $\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$. Each stage is consistent and compose of Conv. Encoder blocks followed by SwiftFormer Encoder. Between two consecutive stages, we incorporate downsampling layer to reduce the spatial size by a factor of two and increase the feature dimensions. **Bottom Row:** We show the design of the Conv. Eencoder (left) and the SwiftFormer Encoder (right). The Conv. Encoder is designed to learn effective local representations and consists of 3×3 depth-wise convolutions followed by two point-wise convolutions for channel mixing. The SwiftFormer Encoder aims to learn enriched local-global representations. It begins with local convolutional layers to extract local features, followed by the efficient additive attention module (see Fig. 2 (d)) and linear layers.

the element-wise product to form global context ($\mathbb{R}^{n\times d}$). This matrix shares a similarity with the attention matrix in MHSA and captures information from every token and has the flexibility to learn the correlation in the input sequence. However, it is comparatively inexpensive to compute compared to MHSA and has linear complexity with the token length. Inspired by the transformer architecture, we employ a linear transformation layer to query-key interactions to learn the hidden representation of the tokens. The output of the efficient additive attention $\hat{\mathbf{x}}$ can be described as:

$$\hat{\mathbf{x}} = \hat{\mathbf{Q}} + \mathbf{T}\Big(\mathbf{K} * \mathbf{q}\Big). \tag{6}$$

where $\hat{\mathbf{Q}}$ denotes to the normalized query matrix, $\mathbf{T}$ denotes to the linear transformation.

3.3. SwiftFormer Architecture

Our SwiftFormer is based on the recently introduced EfficientFormer [20]. The main idea of EfficientFormer is to introduce 4D MetaBlocks based on PoolFormer [53] to learn local representations efficiently, while using 3D MetaBlocks based on self-attention to encode global context. However, the performance of EfficientFormer is limited by two design choices. Firstly, it uses ineffective token mixing, and secondly, it only employs 3D MetaBlocks in the last stage due to quadratic complexity of MHSA. This likely leads to inconsistent and insufficient contextual representation. To address these limitations, our SwiftFormer improves the token mixing by using a simple yet effective Conv. Encoder. Further, we introduce efficient additive attention module that can be incorporated in all stages (Sec. 3.2). This leads to more consistent learning of local-global representations. It is worth mentioning that EfficientFormer employs a latency-driven slimming method to obtain optimal configurations for its model variants, which leads to maximizing the speed. In contrast, our SwiftFormer models are built without using any neural architecture search.

Fig. 3 shows an overview of our proposed architecture. The main components are: **(i)** Effective Conv. Encoder, and **(ii)** SwiftFormer Encoder. In contrast to other hybrid designs, the proposed architecture is consistent and has Conv. Encoders followed by SwiftFormer Encoder in all stages. Our architecture extracts hierarchical features at four different scales across four stages. At the beginning of the network, the input image of size $H\times W\times 3$ is fed through `Patch Embedding` layer, implemented with two 3×3 convolutions with a stride of 2, resulting $\frac{H}{4}\times\frac{W}{4}\times C_1$ feature maps. Then, the output feature maps are fed into the first stage, which begins with Conv. Encoder to extract spatial features, followed by SwiftFormer to learn the local-global information. Between two consecutive stages, there is a downsampling layer to increase the channel dimension and reduce the token length. Next, the resulting feature maps are subsequently fed into the second, third, and fourth stages of the architecture, producing $\frac{H}{8}\times\frac{W}{8}\times C_2$, $\frac{H}{16}\times\frac{W}{16}\times C_3$, and $\frac{H}{32}\times\frac{W}{32}\times C_4$ dimensional feature maps, respectively. Hence, each stage learns local-global features at different scales of the input image, which allows the network to have enriched representation.

Effective Conv. Encoder: The baseline EfficientFormer [20] employs 3×3 average pooling layers as a local token mixer, similar to PoolFormer [53]. Although PoolFormer layers are known for their fast inference speed, replacing them with depth-wise convolutions does not increase the latency. Further, it improves the performance without increasing the parameters and latency. Specifically, the features maps $\mathcal{X}_i$ are fed into 3×3 depth-wise convolution (`DWConv`) followed by Batch Normalization (`BN`). Then, the resulting features are fed into two point-wise convolutions ($\texttt{Conv}_1$) alongside `GeLU` activation. Finally, we incorporate a skip connection to enable information to flow across the network. The Conv. Encoder is defined as:

$$\hat{\mathcal{X}}_i = \texttt{Conv}_1(\texttt{Conv}_{1,G}(\texttt{DWConv}_{BN}(\mathcal{X}_i))) + \mathcal{X}_i. \tag{7}$$

where $\mathcal{X}_i$ refers to the input features, $\texttt{Conv}_{1,G}$ refers to point-wise convolution followed by `GeLU`, $\texttt{DWConv}_{BN}$

refers to depth-wise convolution followed by Batch Normalization, and $\hat{\mathcal{X}}_i$ refers to the output feature maps.

SwiftFormer Encoder: This module is carefully designed to efficiently encode enriched local-global representation in each stage. As shown in Fig. 3, the initial block of the SwiftFormer Encoder is composed of 3×3 depth-wise convolution followed by point-wise convolution, which enables the module to learn spatial information and encode local representation. Then, the resulting feature maps are fed into the efficient additive attention block, which aims to learn contextual information at each scale of the input size. Finally, the output feature maps are fed into a `Linear` block, which composes of two 1×1 point-wise convolution layers, Batch Normalization, and `GeLU` activation to generate non-linear features. The SwiftFormer Encoder is described as:

$$\begin{aligned} \hat{\mathcal{X}}_i &= \texttt{Conv}_1(\texttt{DWConv}_{BN}(\hat{\mathcal{X}}_i)), \\ \hat{\mathcal{X}}_i &= \texttt{QK}(\hat{\mathcal{X}}_i) + \hat{\mathcal{X}}_i, \\ \hat{\mathcal{X}}_{i+1} &= \texttt{Conv}_1(\texttt{Conv}_{BN,1,G}(\hat{\mathcal{X}}_i)) + \hat{\mathcal{X}}_i. \end{aligned} \tag{8}$$

where $\texttt{Conv}_{BN,1,G}$ denotes to Batch Normalization, followed by, 1×1 `Conv` layer, followed by `GeLU`, and `QK` denotes the efficient additive attention (explained in Sec. 6).

4. Experiments

We evaluate our SwiftFormer models across four downstream tasks: classification on ImageNet-1K [7], object detection and instance segmentation on MS-COCO 2017 [21], and semantic segmentation on ADE20K [58].

4.1. Implementation Details

ImageNet-1K [7]: All of our models are trained from scratch on ImageNet-1K dataset for 300 epochs with AdamW optimizer [26] and cosine learning rate scheduler with an initial learning rate of $1e^{-3}$. We use a linear warm-up for 5 epochs. We use an image resolution of 224×224 for both training and testing. Following the training recipe of [20], we use the same teacher model for distillation [36]. The experiments are conducted with PyTorch 1.12 [35] using 8 NVIDIA A100 GPUs. The latency is measured using iPhone 14 (iOS 16), and the throughput is measured using A100 40 GB GPU. For latency measurements, we compile the models using CoreML library [6] and perform inference with a batch size of 1. For the throughput on A100, the inference is performed using a batch size of 128.

MS-COCO 2017 [21]: We use our ImageNet pre-trained models as the backbones in Mask-RCNN framework for object detection and instance segmentation on MS-COCO 2017 dataset. The dataset contains 118K training and 5K validation images. Following [20], we finetune our models for 12 epochs with an image size of 1333×800 and batch size of 32 using AdamW optimizer. We use learning rate of $2e^{-4}$ and report the performance for detection and instance segmentation in terms of mean average precision (mAP).

ADE20K [58]: The dataset comprises 20K training and 2K validation images and contains 150 class categories for scene parsing. Similar to [20] we use our ImageNet pre-trained models to extract image features and semantic FPN [18] as a decoder for segmentation. The model is trained with an image size of 512×512 for 40K iterations with a batch size of 32 using AdamW optimizer. We use poly learning rate scheduling with an initial learning rate of $2e^{-4}$. We report the semantic segmentation performance in terms of mean intersection over union (mIoU).

4.2. Baseline Comparison

Table 1 illustrates the impact of integrating our proposed contributions into the baseline EfficientFormer-L1 [20] model in terms of ImageNet-1K top-1 accuracy and inference speed. The first row shows the results of the baseline model, which only includes the self-attention based transformer block in the final stage of the network and achieves a top-1 accuracy of 79.2% with a latency of 1.1 ms on an iPhone 14 mobile device. The second row replaces the pool mixers in the baseline model with our proposed Conv. Encoder, resulting in an improvement in performance to 79.9% while maintaining the same latency. In the third row, we replace the transformer block in the baseline with our proposed SwiftFormer Encoder built on the efficient additive attention. Although the performance drops by 0.2%, the inference speed improves by 0.1 ms, and the model has linear complexity with the number of tokens. This enables us to integrate the SwiftFormer Encoder into all stages and achieve better performance while maintaining the same inference speed as of baseline (first versus last row).

Method	Latency (ms)	Top-1 (%)
EfficientFormer-L1 (Baseline)	1.1	79.2
+ Replace Pool Mixer by effective Conv. Encoder	1.1	79.9
+ Replace Transformer block by SwiftFormer Encoder	1.0	79.7
+ Incorporate SwiftFormer block across all stages	**1.1**	**80.9**

Table 1: Baseline comparison between our SwiftFormer-L1 and EfficientFormer-L1 [20] on the ImageNet-1K dataset. The latency is measured on iPhone14 Neural Engine.

4.3. Image Classification

Table 2 presents a comparison of our proposed SwiftFormer models (XS, S, L1, and L3) with previous state-of-the-art ConvNets, transformer-based, and hybrid models. We show that our models set new state-of-the-art results, and outperform the recently introduced EfficientFormer [20] and MobileViT-v2 [31] in all model variants. This comprehensive evaluation shows the advantage of our proposed models in terms of both accuracy and latency on mobile devices.

Model	Type	Latency (ms) ↓	Throughput (A100) ↑	Params(M) ↓	GMACs ↓	Neural Search	Top-1(%) ↑
MobileNet-v2×1.0 [39]	ConvNet	0.8	9889	3.5	0.3	✗	71.8
MobileNet-v3-Large×0.75 [15]	ConvNet	0.8	10934	4.0	0.2	✓	73.3
EdgeViT-XXS [34]	Hybrid	1.7	5965	4.1	0.6	✗	74.4
MobileViT-XS [30]	Hybrid	1.5	3707	2.3	0.7	✗	74.8
SwiftFormer-XS	**Hybrid**	**0.7**	**6034**	**3.5**	**0.6**	✗	**75.7**
MobileNet-v2×1.4 [39]	ConvNet	0.9	7447	6.1	0.6	✗	74.7
MobileNet-v3-Large [15]	ConvNet	0.9	10351	5.4	0.3	✓	75.1
EfficientNet-b0 [42]	ConvNet	1.3	8537	5.3	0.4	✓	77.1
DeiT-T [44]	Transformer	3.8	5860	5.7	3.8	✗	72.2
EdgeViT-XS [34]	Hybrid	2.7	4812	6.7	1.1	✗	77.5
MobileViT-v2×1.0 [31]	Hybrid	1.7	3201	4.9	1.8	✗	78.1
SwiftFormer-S	**Hybrid**	**0.8**	**5051**	**6.1**	**1.0**	✗	**78.5**
MobileFormer-508M [3]	Hybrid	3.0	4443	14.0	0.5	✗	79.3
PoolFormer-S12 [53]	Pool	1.2	3227	12.0	1.8	✗	77.2
EdgeViT-S [34]	Hybrid	3.5	4256	13.1	1.9	✗	81.0
EfficientFormer-L1 [20]	Hybrid	1.1	5046	12.3	1.3	✓	79.2
MobileViT-v2×1.5 [31]	Hybrid	3.4	2356	10.6	4.0	✗	80.4
SwiftFormer-L1	**Hybrid**	**1.1**	**4469**	**12.1**	**1.6**	✗	**80.9**
ResNet-50 [13]	ConvNet	1.9	4835	25.5	4.1	✗	78.5
PoolFormer-S36 [53]	Pool	2.8	1114	31.0	5.0	✗	81.4
ConvNeXt-T [25]	ConvNet	2.5	3235	28.6	4.5	✗	82.1
DeiT-S [44]	Transformer	9.9	2990	22.5	4.5	✗	81.8
Swin-T [25]	Transformer	NA	2635	28.3	4.5	✗	81.3
MobileViT-v2×2.0 [31]	Hybrid	5.0	1906	18.5	7.5	✗	81.2
EfficientFormer-L3 [20]	Hybrid	2.0	2691	31.3	3.9	✓	82.4
SwiftFormer-L3	**Hybrid**	**1.9**	**2890**	**28.5**	**4.0**	✗	**83.0**

Table 2: **Comparison of our proposed SwiftFormer with the state-of-the-art counterpart models on ImgeNet-1K.** The latency is measured on iPhone 14 Neural Engine (iOS 16) and the throughput is measured on Nvidia A100 GPU. Our models run faster than MobileNets, Hybrid, and Transformer models, with a better trade-off between accuracy and model complexity. The error for the latency measurement is less than ± 0.1 ms. The input size remains consistent at 224×224 for all models, except for the MobileViT [30] and MobileViT-v2 [31] variants, which employ a larger dimension of 256×256. Our results are shown in bold for all model variants.

Comparison with ConvNets: Our SwiftFormer models surpass the widely used lightweight CNNs counterparts significantly in terms of top-1 accuracy, while running faster than the highly optimized MobileNet-v2 and MobileNet-v3 on an iPhone 14 mobile device. Specifically, our SwiftFormer-XS runs 0.1 ms faster than MobileNet-v2×1.0 and MobileNet-v3-Large×0.75 and achieve better top-1 accuracy with a margin of 3.9% and 2.4% respectively. Our SwiftFormer-S runs faster than EfficientNet-b0 [42] by 1.6× and achieves 1.4% higher top-1 accuracy. Further, our SwiftFormer-L3 achieves 4.5% and 0.9% gain in top-1 accuracy over ResNet-50 and ConvNeXt-T, respectively, while running at the same latency as ResNet-50 and 1.3× faster than ConvNeXt-T. This demonstrates that our SwiftFormer models, powered by our proposed efficient additive attention, run faster than the lightweight CNN models on mobile devices and achieve superior performance. Recent device-level optimizations for CNN-based models, such as dedicated hardware implementations for convolutions with batch normalization and non-linearity, likely contribute to the high throughput of fully CNN-based models on A100.

Comparison with transformer models: Although transformer models usually outperform CNN-based models in terms of accuracy, they tend to suffer from high latency when running on resource-constrained mobile devices. For instance, DeiT-S, which has a similar model size to ResNet-50 and achieves higher top-1 accuracy by 3.3%, but ResNet-50 runs approximately 5.2× faster on an iPhone 14 mobile device. In contrast, our SwiftFormer-L3 model achieves 1.2% higher accuracy than DeiT-S, while running at the same speed as ResNet-50. Further our SwiftFormer-S model runs approximately 4.7× faster than DeiT-T on an iPhone 14 mobile device and has 6.3% better accuracy.

Comparison with hybrid models: Although most existing hybrid approaches achieve higher accuracy compared to their lightweight CNN counterparts, they still underperform the fully CNN-based models in terms of latency due to the quadratic complexity of multi-head self-attention. For example, EdgeViT-XXS runs at approximately 2× slower compared to MobileNet-v3-Large×0.75. On the other hand, our SwiftFormer-XS has better latency as compared to lightweight CNNs and approximately is 2× faster than EdgeViT-XXS and MobileViT-XS, with an overall 1.3% and 0.9% higher top-1 accuracy respectively. Further, our SwiftFormer-L1 model is 3× faster than the state-of-the-art MobileViT-v2×1.5 with 0.5% better top-1 ac-

Backbone	Detection & Instance Segmentation						Semantic
	$\mathbf{AP}^{box}$	$\mathbf{AP}^{box}_{50}$	$\mathbf{AP}^{box}_{75}$	$\mathbf{AP}^{mask}$	$\mathbf{AP}^{mask}_{50}$	$\mathbf{AP}^{mask}_{75}$	mIoU(%)
ResNet18 [13]	34.0	54.0	36.7	31.2	51.0	32.7	32.9
PoolFormer-S12 [53]	37.3	59.0	40.1	34.6	55.8	36.9	37.2
EfficientFormer-L1 [20]	37.9	60.3	41.0	35.4	57.3	37.3	38.9
SwiftFormer-L1	**41.2**	**63.2**	**44.8**	**38.1**	**60.2**	**40.7**	**41.4**
ResNet50 [13]	38.0	58.6	41.4	34.4	55.1	36.7	36.7
PVT-S [48]	40.4	62.9	43.8	37.8	60.1	40.3	39.9
PoolFormer-S24 [53]	40.1	62.2	43.4	37.0	59.1	39.6	40.3
Swin-T [24]	42.2	64.4	46.2	39.1	64.6	42.0	41.5
EfficientFormer-L3 [20]	41.4	63.9	44.7	38.1	61.0	40.4	43.5
SwiftFormer-L3	**42.7**	**64.4**	**46.7**	**39.1**	**61.7**	**41.8**	**43.9**

Table 3: **Results using SwiftFormer as a backbone on dense prediction tasks**: Object detection and instance segmentation on COCO, whereas semantic segmentation on ADE20K. Our approach outperforms EfficientFormer on all three tasks.

Figure 4: **Results on COCO.** Examples for object detection and instance segmentation on the COCO 2017 validation set. Our SwiftFormer-L1 model can accurately detect and segment instances in images.

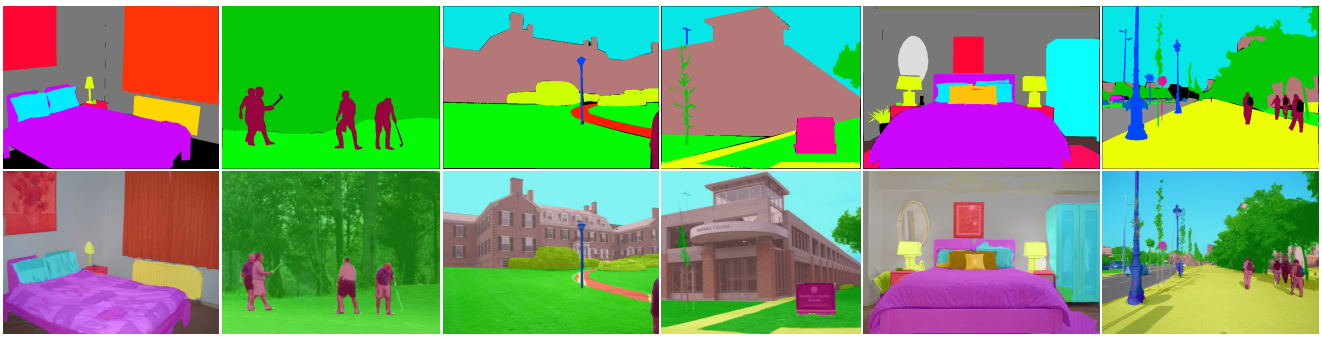

Figure 5: **Qualitative results on ADE20K.** The qualitative examples for semantic segmentation on the ADE20K validation set. **Top:** Ground truth masks. **Bottom:** The semantic segmentation results. Our model can accurately segment various indoor and outdoor scenes.

curacy. Our SwiftFormer-L3 model achieves 83.0% top-1 accuracy and runs at 1.9 ms, which is 2.6× faster than MobileViT-v2×2.0 with an absolute 1.8% accuracy gain.

Knowledge distillation (KD): It is worth mentioning that we use the same teacher model of DeiT [44] and EfficientFormer [20] in Table 2. We perform a baseline comparison to show the efficiency of our method without using KD. EfficientFormer-L1 achieves 77.3% accuracy w/o KD, compared to 79.2% with KD. Our SwiftFormer-L1 achieves 79.8% w/o KD, compared to 80.9% with KD. This shows that our approach achieves a higher absolute gain of 2.5% over EfficientFormer-L1, when not using KD. Our SwiftFormer-XS and SwiftFormer-S w/o KD achieve 74.1% and 77.0%, respectively.

4.4. Object Detection and Instance Segmentation

Table 3 compares the object detection and instance segmentation results of Mask-RCNN with different lightweight backbones. Our SwiftFormer-L1 backbone achieves 41.2 AP box, surpassing the lightweight ResNet18 and PoolFormer-S12 backbones by 7.2 and 3.9 points respectively. Further, it performs better than the previous state-of-the-art EfficientFormer-L1 backbone by 3.3 AP box. For instance segmentation, our method achieves 38.1 AP mask

score which is 2.7 points better than the previous state-of-the-art. Similar trend is observed for SwinftFormer-L3 backbone, which surpasses the previous state-of-the-art EfficientFormer-L3 backbone by 1.3 points and 1.0 points in AP box and mask respectively. The improvement in the downstream detection and instance segmentation tasks illustrates the effectiveness of our SwiftFormer backbone models for the dense prediction tasks.

4.5. Semantic Segmentation

Table 3 shows the semantic segmentation results of SwiftFormer backbone-based models as compared to previously proposed backbones. We achieve 41.4% mean intersection over union score using SwiftFormer-L1, surpassing ResNet18 by 8.5%, PoolFormer-S12 by 4.2%, and the state-of-the-art EfficientFormer-L1 by 2.5%. Similarly, our SwiftFormer-L3 backbone-based segmentation models achieve 43.9 mIoU, surpassing all previous methods.

4.6. Qualitative Results

Fig. 4 shows the qualitative results of our detection and instance segmentation model on COCO dataset. We also show in Fig. 5 qualitative examples for semantic segmentation results on the ADE20K dataset. Additionally, we demonstrate in Fig. 6 a baseline comparison with the baseline EfficentFormer-L1. The baseline misclassifies a bird as a sheep in the first example and as a bear in the second example, while misclassifies kites as birds in the third example. Our SwiftFormer-L1 accurately detects and segments objects in these examples, which demonstrates the ability of SwiftFormer to accurately identify and segment objects within a diverse range of indoor and outdoor settings.

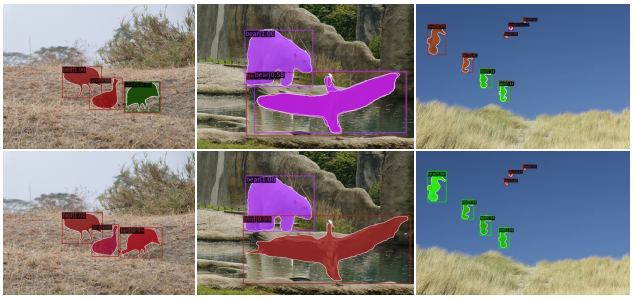

Figure 6: Qualitative comparison between EfficientFormer-L1 (**top**) and our SwiftFormer-L1 (**bottom**) on images from COCO val set for detection and instance segmentation.

5. Ablation study

To show the effectiveness of the proposed efficient additive attention, we conduct a comparison with other attention variants over SwiftFormer-L1 in Table. 4. It is notable that the proposed efficient additive attention achieves the best trade-off between latency and top-1 accuracy. To demonstrate further insights on the interpretability of the proposed efficient additive attention, we visualize the learned global attention query vector α of Eq. 4 of the first and last stages of the network. The global query vector of the efficient additive attention aligns with the semantics of the image and hence acts as an informative source in the additive attention mechanism. Also, it localizes the features of the objects accurately with fine details in the first stage.

Attention Method	Latency (ms)↓	Throughput↑	Top-1 (%)↑
Self-attention (Heads=2)	8.5	3316	80.6
Self-attention (Heads=4)	18.0	3035	81.1
Transpose self-attention (Heads=2)	2.0	3671	80.2
Separable self-attention	1.4	3940	80.6
Efficient Additive Attention (Ours)	1.1	4469	80.9

Table 4: **Quantitative analysis of Fig. 2**. We replace our attention module with these methods in the same setting.

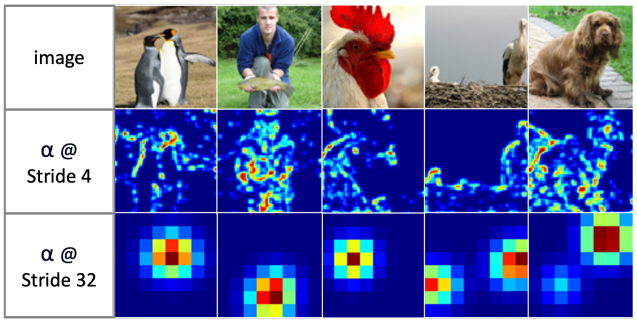

Figure 7: Visualization of the learned global query vector of the efficient additive attention from SwiftFormer-L1 model at different stages. Both the first and last stages show that the global context is captured effectively.

6. Conclusion

Transformers have gained popularity in vision applications due to their effective use of self-attention computation. However, their use in mobile vision applications is challenging due to the quadratic nature of self-attention, which is computationally expensive on mobile devices. In this work, we propose a novel efficient additive attention that replaces the expensive matrix multiplication operations with element-wise operations and eliminates explicit keys-values interaction. Our proposed attention is linear with respect to the input tokens and used in all stages of the network.

Although we achieve a superior balance between accuracy and efficiency, we observe that in the case of dense small object detection, our approach is less accurate likely due to not using positional encoding or attention biases. Our future work is to improve the performance of efficient additive attention in dense prediction tasks.

References

[1] Adrian Bulat and Georgios Tzimiropoulos. Bit-mixer: Mixed-precision networks with runtime bit-width selection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, 2020.

[3] 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*, 2022.

[4] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[5] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *Advances in Neural Information Processing Systems*, 2021.

[6] CoreMLTools. Use coremltools to convert models from third-party libraries to core ml., 2021.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] 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*, 2021.

[10] Mohsen Fayyaz, Soroush Abbasi Kouhpayegani, Farnoush Rezaei Jafari, Eric Sommerlade, Hamid Reza Vaezi Joze, Hamed Pirsiavash, and Juergen Gall. Adaptive token sampling for efficient vision transformers. 2022.

[11] TSong Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*, 2016.

[12] Ali Hatamizadeh, Hongxu Yin, Greg Heinrich, Jan Kautz, and Pavlo Molchanov. Global context vision transformers. In *International Conference on Machine Learning*, pages 12633–12646. PMLR, 2023.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

[14] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. In *ICCV*, 2021.

[15] 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 mobilenet-v3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[16] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[17] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*, 2018.

[18] A. Kirillov, R. Girshick, K. He, and P. Dollar. Panoptic feature pyramid networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[19] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

[20] Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. 2022.

[21] 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 *European Conference on Computer Vision*, 2014.

[22] Yahui Liu, Enver Sangineto, Wei Bi, Nicu Sebe, Bruno Lepri, and Marco De Nadai. Efficient training of visual transformers with small datasets. In *Advances in Neural Information Processing Systems*, 2021.

[23] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[25] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[27] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *European Conference on Computer Vision*, 2018.

[28] Muhammad Maaz, Hanoona Rasheed, Salman Khan, Fahad Shahbaz Khan, Rao Muhammad Anwer, and Ming-Hsuan Yang. Class-agnostic object detection with multi-modal transformer. In *European Conference on Computer Vision*, 2022.

[29] Muhammad Maaz, Abdelrahman Shaker, Hisham Cholakkal, Salman Khan, Syed Waqas Zamir, Rao Muhammad Anwer, and Fahad Shahbaz Khan. Edgenext: Efficiently amalgamated cnn-transformer architecture for mobile vision applications. In *CADL2022*, 2022.

[30] Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. In *International Conference on Learning Representations*, 2022.

[31] Sachin Mehta and Mohammad Rastegari. Separable self-attention for mobile vision transformers. *Transactions on Machine Learning Research*, 2023.

[32] Sachin Mehta, Mohammad Rastegari, Linda Shapiro, and Hannaneh Hajishirzi. 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.

[33] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional detr for fast training convergence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

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

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

[36] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, 2020.

[37] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems*, 2021.

[38] Michael Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: Adaptive space-time tokenization for videos. In *Advances in Neural Information Processing Systems*, 2021.

[39] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenet-v2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[40] Abdelrahman Shaker, Muhammad Maaz, Hanoona Rasheed, Salman Khan, Ming-Hsuan Yang, and Fahad Shahbaz Khan. Unetr++: Delving into efficient and accurate 3d medical image segmentation. *arXiv:2212.04497*, 2022.

[41] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[42] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[43] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. Mlp-mixer: An all-mlp architecture for vision. In *Advances in Neural Information Processing Systems*, 2021.

[44] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, 2021.

[45] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *ECCV*, 2022.

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

[47] 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 *CVPR*, 2021.

[48] 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.

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

[50] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, 2016.

[51] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. Fastformer: Additive attention can be all you need. *arXiv preprint arXiv:2108.09084*, 2021.

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

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

[54] Haokui Zhang, Wenze Hu, and Xiaoyu Wang. Edgeformer: Improving light-weight convnets by learning from vision transformers. *arXiv preprint arXiv:2203.03952*, 2022.

[55] Haokui Zhang, Wenze Hu, and Xiaoyu Wang. Parc-net: Position aware circular convolution with merits from convnets and transformer. In *European Conference on Computer Vision*, 2022.

[56] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Harry Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *International Conference on Learning Representations*, 2022.

[57] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[58] 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.

[59] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021.