International Journal of Computer Vision (2025) 133:3645–3666
https://doi.org/10.1007/s11263-025-02345-2

SeaFormer++: Squeeze-Enhanced Axial Transformer for Mobile Visual Recognition

Qiang Wan[1] · Zilong Huang[2] · Jiachen Lu[1] · Gang Yu[3] · Li Zhang[1]

Received: 7 May 2024 / Accepted: 2 January 2025 / Published online: 25 January 2025

Abstract

Since the introduction of Vision Transformers, the landscape of many computer vision tasks (e.g., semantic segmentation), which has been overwhelmingly dominated by CNNs, recently has significantly revolutionized. However, the computational cost and memory requirement renders these methods unsuitable on the mobile device. In this paper, we introduce a new method squeeze-enhanced Axial Transformer (SeaFormer) for mobile visual recognition. Specifically, we design a generic attention block characterized by the formulation of squeeze Axial and detail enhancement. It can be further used to create a family of backbone architectures with superior cost-effectiveness. Coupled with a light segmentation head, we achieve the best trade-off between segmentation accuracy and latency on the ARM-based mobile devices on the ADE20K, Cityscapes Pascal Context and COCO-Stuff datasets. Critically, we beat both the mobile-friendly rivals and Transformer-based counterparts with better performance and lower latency without bells and whistles. Furthermore, we incorporate a feature upsampling-based multi-resolution distillation technique, further reducing the inference latency of the proposed framework. Beyond semantic segmentation, we further apply the proposed SeaFormer architecture to image classification and object detection problems, demonstrating the potential of serving as a versatile mobile-friendly backbone. Our code and models are made publicly available at https://github.com/fudan-zvg/SeaFormer.

Keywords Transformer · Semantic segmentation · Knowledge distillation

1 Introduction

As a fundamental problem in computer vision, semantic segmentation aims to assign a semantic class label to each pixel in an image. Conventional methods rely on stacking local convolution kernel (Long et al., 2015) to perceive the long-range structure information of the image. Since the introduction of Vision Transformers (Dosovitskiy et al., 2021), the landscape of semantic segmentation has significantly revolutionized. Transformer-based approaches (Zheng et al., 2021; Xie et al., 2021) have remarkably demonstrated the capability of global context modeling. However, the computational cost and memory requirement of Transformer render these methods unsuitable on mobile devices, especially for high-resolution imagery inputs. Following conventional wisdom of efficient operation, local/window-based attention (Luong, 2015; Liu et al., 2021; Huang et al., 2021; Yuan et al., 2021), Axial attention (Huang et al., 2019; Ho et al., 2019; Wang et al., 2020), dynamic graph message passing (Zhang et al., 2020, 2023) and some lightweight attention mechanisms (Hou et al., 2020; Li et al., 2021a, b, 2020; Liu et al., 2018; Shen et al., 2021; Xu et al., 2021; Cao et al., 2019; Woo et al., 2018; Wang et al., 2020; Choromanski et al., 2021; Chen, 2017; Mehta and Rastegari, 2022) are introduced.

However, these advances are still insufficient to satisfy the design requirements and constraints for mobile devices due to the high latency on the high-resolution inputs (see Fig. 1). Recently there is a surge of interest in building a Transformer-based semantic segmentation. In order to reduce the computation cost at high resolution, TopFormer (Zhang et al., 2022) dedicates to applying the global attention at a 1/64 scale of the original input, which definitely

Communicated by Kaiyang Zhou.

✉ Li Zhang
lizhangfd@fudan.edu.cn

[1] School of Data Science, Fudan University, Yangpu, 200433 Shanghai, China

[2] ByteDance, 1 Raffles Quay, #26-10, 048583 Singapore, Singapore

[3] Tencent, Xuhui, Shanghai 200233, Shanghai, China

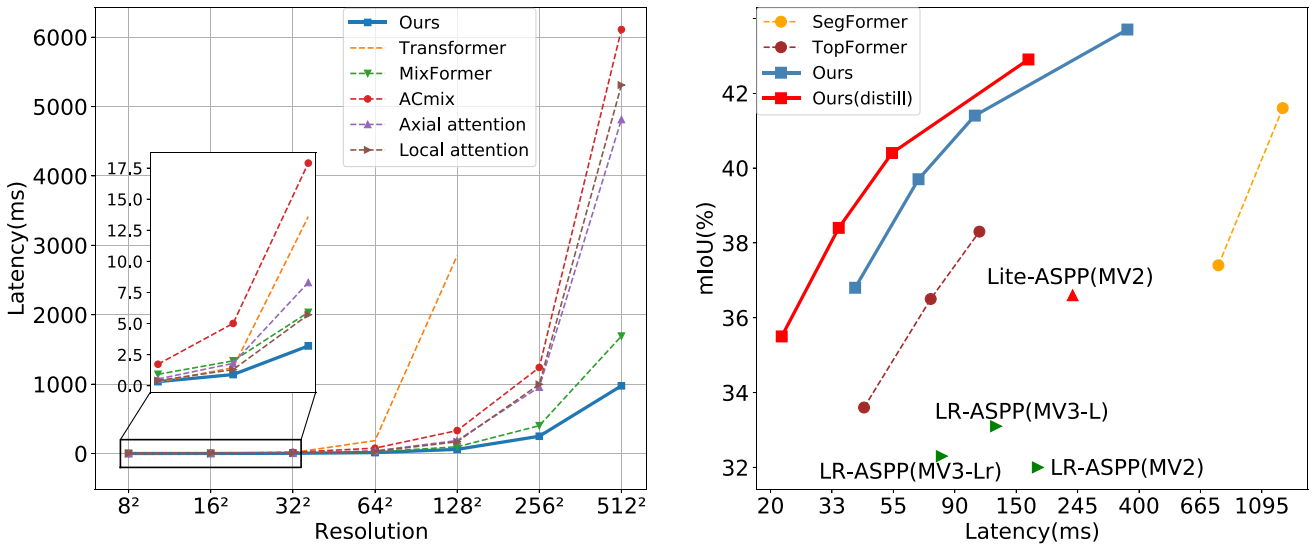


Fig. 1 ***Left***: Latency comparison with Transformer (Vaswani, 2017), MixFormer (Chen et al., 2022), ACmix (Pan et al., 2022), Axial attention (Ho et al., 2019) and local attention (Luong, 2015). It is measured with a single module of channel dimension 64 on a Qualcomm Snapdragon 865 processor. ***Right***: The mIoU versus latency on the ADE20K *val* set. MV2 means MobileNetV2 (Sandler et al., 2018). MV3-L means MobileNetV3-Large (Howard et al., 2019). MV3-Lr denotes MobileNetV3-Large-reduce (Howard et al., 2019). The latency is measured on a single Qualcomm Snapdragon 865, and only an ARM CPU core is used for speed testing. No other means of acceleration, e.g., GPU or quantification, is used. For figure *Right*, the input size is 512×512. SeaFormer achieves superior trade-off between mIoU and latency

harms the segmentation performance. To solve the dilemma of high-resolution computation for pixel-wise segmentation task and low latency requirements on the mobile device in a performance-harmless way, we propose a family of mobile-friendly Transformer-based semantic segmentation models, dubbed squeeze-enhanced Axial Transformer (SeaFormer), which reduces the computational complexity of axial attention from $\mathcal{O}((H+W)HW)$ to $\mathcal{O}(HW)$, to achieve superior accuracy-efficiency trade-off on mobile devices and fill the vacancy of mobile-friendly efficient Transformer.

The core building block *squeeze-enhanced Axial attention* (SEA attention) seeks to squeeze (pool) the input feature maps along the horizontal/vertical axis into a compact column/row and computes self-attention. We concatenate query, keys and values to compensate the detail information sacrificed during squeeze and then feed it into a depth-wise convolution layer to enhance local details. The SEA attention module is a novel enhancement designed to reduce computational complexity, making it suitable for mobile devices without sacrificing performance. Coupled with a light segmentation head and several lightweight fusion blocks, our design (see Fig. 2) with the proposed SeaFormer layer in the small-scale feature is capable of conducting high-resolution image semantic segmentation with low latency on the mobile device. Our dual-branch network architecture incorporates these unique design elements that optimize efficiency and accuracy for mobile applications. As shown in Fig. 1, the proposed SeaFormer outperforms other efficient neural networks on the ADE20K dataset with lower latency. In particular, SeaFormer-Base is superior to the lightweight CNN counterpart MobileNetV3 (41.0 *vs.*33.1 mIoU) with lower latency (106ms *vs.*126ms) on an ARM-based mobile device. Furthermore, we have proposed a multi-resolution distillation framework that further enhances the model's efficiency and accuracy.

We make the following **contributions**: **(i)** We introduce a novel squeeze-enhanced Axial Transformer (SeaFormer) framework for mobile semantic segmentation; **(ii)** Critically, we design a generic attention block characterized by the formulation of squeeze Axial and detail enhancement; It can be used to create a family of backbone architectures with superior cost-effectiveness; **(iii)** We show top performance on the ADE20K and Cityscapes datasets, beating both the mobile-friendly rival and Transformer-based segmentation model with clear margins; **(iv)** Beyond semantic segmentation, we further apply the proposed SeaFormer architecture to the image classification problem, demonstrating the potential of serving as a versatile mobile-friendly backbone.

A preliminary version of this work was presented in ICLR 2023 (Wan et al., 2023). In this paper, we have further extended our conference version as follows: **(i)** We propose an adaptive squeeze and expand method in Squeeze-axial attention, using a learnable mask to map all tokens of query/key/values to a single token in each row and column; **(ii)** We present a feature upsampling based multi-resolution

distillation approach, further reducing the inference latency of the proposed framework.

2 Related Work

2.1 Combination of Transformers and Convolution

Convolution is relatively efficient but not suitable to capture long-range dependencies and vision Transformer has the powerful capability with a global receptive field but lacks efficiency due to the computation of self-attention. In order to make full use of both of their advantages, MobileViT (Mehta and Rastegari, 2022), TopFormer (Zhang et al., 2022), LVT (Yang et al., 2022), Mobile-Former (Chen et al., 2022), EdgeViTs (Pan et al., 2022), MobileViTv2 (Mehta and Rastegari, 2022), EdgeFormer (Zhang et al., 2022) and EfficientFormer (Li et al., 2022) are constructed as efficient ViTs by combining convolution with Transformers. MobileViT, Mobile-Former, TopFormer and EfficientFormer are restricted by Transformer blocks and have to trade off between efficiency and performance in model design. LVT, MobileViTv2 and EdgeViTs keep the model size small at the cost of relatively high computation, which also means high latency.

2.2 Axial Attention and Variants

Axial attention (Huang et al., 2019; Ho et al., 2019; Wang et al., 2020) is designed to reduce the computational complexity of original global self-attention (Vaswani, 2017). It computes self-attention over a single axis at a time and stacks a horizontal and a vertical axial attention module to obtain the global receptive field. Strip pooling (Hou et al., 2020) and Coordinate attention (Hou et al., 2021) uses a band shape pooling window to pool along either the horizontal or the vertical dimension to gather long-range context. Kronecker Attention Networks (Gao et al., 2020) uses the juxtaposition of horizontal and lateral average matrices to average the input matrices and performs attention operation. These methods and other similar implementations provide performance gains partly at considerably low computational cost compared with Axial attention. However, they ignore the lack of local details brought by the pooling/average operation.

2.3 Mobile Semantic Segmentation

The mainstream of efficient segmentation methods are based on lightweight CNNs. DFANet (Li et al., 2019) adopts a lightweight backbone to reduce computation cost and adds a feature aggregation module to refine high-level and low-level features. ICNet (Zhao et al., 2018) designs an image cascade network to speed up the algorithm, while BiSeNet (Yu et al., 2018, 2021) proposes two-stream paths for low-level details and high-level context information, separately. Fast-SCNN (Poudel et al., 2019) shares the computational cost of the multi-branch network to yield a run-time fast segmentation CNN. TopFormer (Zhang et al., 2022) presents a new architecture with a combination of CNNs and ViT and achieves a good trade-off between accuracy and computational cost for mobile semantic segmentation. However, it is still restricted by the heavy computation load of global self-attention.

2.4 Knowledge Distillation

Knowledge Distillation (KD) is a widely used technique in model compression, where a smaller model (student) is trained to mimic the behavior of a larger, well-performing model (teacher). The concept was first introduced by Hinton (2015), where the student model learns from the teacher's softened output probabilities. Since then, various distillation methods have been proposed (Romero et al., 2014; Park et al., 2019; Tian et al., 2019; Liu et al., 2022; He et al., 2019; Liu et al., 2019; Yang et al., 2022; Cho & Hariharan, 2019; Heo et al., 2019; Kim et al., 2018; Li et al., 2023, 2021; Mirzadeh et al., 2020; Yang et al., 2019; Yim et al., 2017; Zhang et al., 2018; Zhao et al., 2022; Wang et al., 2024) to improve the learning process, focusing on feature-based distillation (Romero et al., 2014), relational knowledge distillation (Park et al., 2019) and contrastive and adversarial approaches to distillation (Tian et al., 2019). Our work builds on these great works by proposing a multi-resolution distillation method that aligns features at different feature resolutions between teacher and student, enabling a more efficient and effective knowledge transfer in dense prediction tasks, such as segmentation.

3 Method

3.1 Overall Architecture

Inspired by the two-branch architectures (Yu et al., 2021; Poudel et al., 2019; Hong et al., 2021; Huang et al., 2021; Chen et al., 2022), we design a squeeze-enhanced Axial Transformer (SeaFormer) framework. As is shown in Fig. 2, SeaFormer consists of these parts: *shared STEM*, *context branch*, *spatial branch*, *fusion block* and *light segmentation head*. For a fair comparison, we follow TopFormer (Zhang et al., 2022) to design the STEM. It consists of one regular convolution with stride of 2 followed by four MobileNet blocks where stride of the first and third block is 2. The context branch and the spatial branch share the produced feature map, which allows us to build a fast semantic segmentation model.

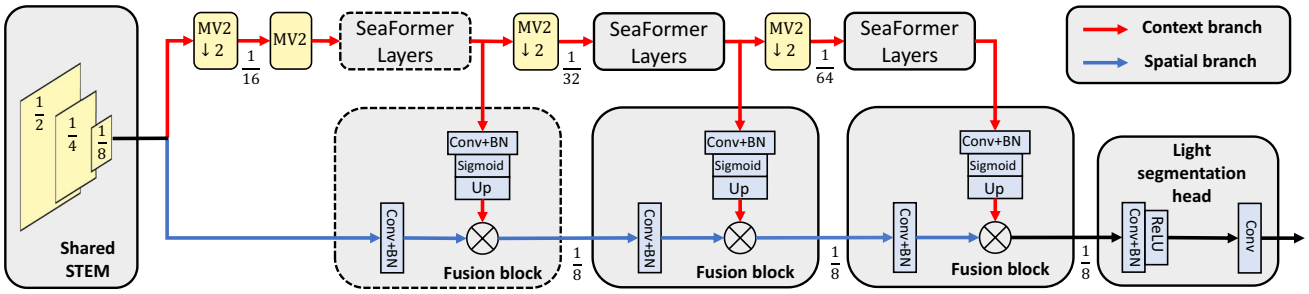


Fig. 2 The overall architecture of SeaFormer. It contains shared STEM, context branch (**red**), spatial branch (**blue**), fusion block and light segmentation head. `MV2` block means MobileNetV2 block and `MV2` ↓2 means MobileNetV2 block with downsampling. SeaFormer layers and fusion block with dash box only exist in SeaFormer-L. The symbol ⊗ denotes element-wise multiplication

3.1.1 Context Branch

The context branch is designed to capture context-rich information from the feature map $\mathbf{x}_s$. As illustrated in the red branch of Fig. 2, the context branch is divided into three stages. To obtain larger receptive field, we stack SeaFormer layers after applying a MobileNet block to down-sampling and expanding feature dimension. Compared with the standard convolution as the down-sampling module, MobileNet block increases the representation capacity of the model while maintaining a lower amount of computation and latency. For variants except SeaFormer-Large, SeaFormer layers are applied in the last two stages for superior trade-off between accuracy and efficiency. For SeaFormer-Large, we insert SeaFormer layers in each stage of context branch. To achieve a good trade-off between segmentation accuracy and inference speed, we design a squeeze-enhanced Axial attention block (SEA attention) illustrated in the next subsection.

3.1.2 Spatial Branch

The spatial branch is designed to obtain spatial information in high resolution. Identical to the context branch, the spatial branch reuses feature maps $\mathbf{x}_s$. However, the feature from the early convolution layers contains rich spatial details but lacks high-level semantic information. Consequently, we design a fusion block to fuse the features in the context branch into the spatial branch, bringing high-level semantic information into the low-level spatial information.

3.1.3 Fusion Block

As depicted in Fig. 2, high resolution feature maps in the spatial branch are followed by a 1×1 convolution and a batch normalization layer to produce a feature to fuse. Low resolution feature maps in the context branch are fed into a 1×1 convolution layer, a batch normalization layer, a sigmoid layer and up-sampled to high resolution to produce semantics weights by bilinear interpolation. Then, the semantics weights from context branch are element-wisely multiplied to the high resolution feature from spatial branch. The fusion block enables low-level spatial features to obtain high-level semantic information.

3.1.4 Light Segmentation Head

The feature after the last fusion block is fed into the proposed segmentation head directly, as demonstrated in Fig. 2. For fast inference purpose, our light segmentation head consists of two convolution layers, which are followed by a batch normalization layer separately and the feature from the first batch normalization layer is fed into an activation layer (Fig. 3).

3.2 Squeeze-Enhanced Axial Attention

The global attention can be expressed as

$$\mathbf{y}_o = \sum_{p \in \mathcal{G}(o)} \text{softmax}_p \left(\mathbf{q}_o^\top \mathbf{k}_p \right) \mathbf{v}_p \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$. $\mathbf{q}, \mathbf{k}, \mathbf{v}$ are linear projection of $\mathbf{x}$, *i.e.*$\mathbf{q} = \mathbf{W}_q \mathbf{x}, \mathbf{k} = \mathbf{W}_k \mathbf{x}, \mathbf{v} = \mathbf{W}_v \mathbf{x}$, where $\mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{C_{qk} \times C}, \mathbf{W}_v \in \mathbb{R}^{C_v \times C}$ are learnable weights. $\mathcal{G}(o)$ means all positions on the feature map of location $o = (i, j)$. When traditional attention module is applied on a feature map of $H \times W \times C$, the time complexity can be $\mathcal{O}(H^2 W^2 (C_{qk} + C_v))$, leading to low efficiency and high latency.

$$\mathbf{y}_o = \sum_{p \in \mathcal{N}_{m \times m}(o)} \text{softmax}_p \left(\mathbf{q}_o^\top \mathbf{k}_p \right) \mathbf{v}_p \tag{2}$$

$$\begin{aligned}\mathbf{y}_o = &\sum_{p\in\mathcal{N}_{1\times W}(o)} \text{softmax}_p\left(\mathbf{q}_o^\top \mathbf{k}_p\right)\mathbf{v}_p \\ &+ \sum_{p\in\mathcal{N}_{H\times 1}(o)} \text{softmax}_p\left(\mathbf{q}_o^\top \mathbf{k}_p\right)\mathbf{v}_p\end{aligned} \tag{3}$$

To improve the efficiency, there are some works (Liu et al., 2021; Huang et al., 2019; Ho et al., 2019) computing self-attention within the local region. We show two most representative efficient Transformer in Eqs. 2, 3. Equation 2 is represented by window-based attention (Luong, 2015) successfully reducing the time complexity to $\mathcal{O}(m^2HW(C_{qk}+C_v)) = \mathcal{O}(HW)$, where $\mathcal{N}_{m\times m}(o)$ means the neighbour $m \times m$ positions of o, but loosing global receptiveness. The Eq. 3 is represented by Axial attention (Ho et al., 2019), which only reduces the time complexity to $\mathcal{O}((H+W)HW(C_{qk}+C_v)) = \mathcal{O}((HW)^{1.5})$, where $\mathcal{N}_{H\times 1}(o)$ means all the positions of the column of o; $\mathcal{N}_{1\times W}(o)$ means all the positions of the row of o.

According to their drawbacks, we propose the mobile-friendly squeeze-enhanced Axial attention, with a succinct squeeze Axial attention for global semantics extraction and an efficient convolution-based detail enhancement kernel for local details supplement.

$$\begin{aligned}\mathbf{q}_{(h)} &= \frac{1}{W}\left(\mathbf{q}^{\to(H,C_{qk},W)}\mathbf{A}_W^{\to(H,W,1)}\right)^{\to(H,C_{qk})} \\ \mathbf{q}_{(v)} &= \frac{1}{H}\left(\mathbf{q}^{\to(W,C_{qk},H)}\mathbf{A}_H^{\to(W,H,1)}\right)^{\to(W,C_{qk})}\end{aligned} \tag{4}$$

3.2.1 Squeeze Axial Attention

To achieve a more efficient computation and aggregate global information at the same time, we resort to a more radical strategy. In the same way, $\mathbf{q}, \mathbf{k}, \mathbf{v}$ are first get from $\mathbf{x}$ with $\mathbf{W}_q^{(s)}, \mathbf{W}_k^{(s)} \in \mathbb{R}^{C_{qk}\times C}$, $\mathbf{W}_v^{(s)} \in \mathbb{R}^{C_v\times C}$. According to Eq. 4, we first implement *horizontal squeeze* by employing an input adaptive approach, using a learnable mask to map all tokens of query to a single token in each row. In the same way, the second row of the equation shows the *vertical squeeze* in the vertical direction. $\mathbf{z}^{\to(\cdot)}$ means permuting the dimension of tensor $\mathbf{z}$ as given, and $\mathbf{A}$ is a learnable mask that is adaptively adjusted according to the input feature $\mathbf{x}$. It is formed by applying a 1×1 convolution and batch normalization layer on the input feature map $\mathbf{x}$ The adaptive squeeze operation on $\mathbf{q}$ also repeats on $\mathbf{k}$ and $\mathbf{v}$, so we finally get $\mathbf{q}_{(h)}, \mathbf{k}_{(h)}, \mathbf{v}_{(h)} \in \mathbb{R}^{H\times C_{qk}}$, $\mathbf{q}_{(v)}, \mathbf{k}_{(v)}, \mathbf{v}_{(v)} \in \mathbb{R}^{W\times C_{qk}}$. The squeeze operation reserves the global information to a single axis in an adaptive manner, thus greatly alleviating the following global semantic extraction shown by Eq. 5.

$$\begin{aligned}\mathbf{y}_{(i,j)} = &\sum_{p=1}^{H} \text{softmax}_p\left(\mathbf{q}_{(h)i}^\top \mathbf{k}_{(h)p}\right)\mathbf{v}_{(h)p} \\ &+ \sum_{p=1}^{W} \text{softmax}_p\left(\mathbf{q}_{(v)j}^\top \mathbf{k}_{(v)p}\right)\mathbf{v}_{(v)p}\end{aligned} \tag{5}$$

Each position of the feature map propagates information only on two squeezed axial features. Similar to adaptive squeezing operation, a 1×1 convolution and batch normalization layer is used to generate an input-adaptive mask for feature restoration. The detail is shown in Fig. 4. Compared with the pooling operation for squeezing and broadcast for expanding, the adaptive squeezing and expanding operations help the model aggregate spatial information in an input-adaptive way without introducing excessive computational overhead. The empirical study confirms the effectiveness of the approach. Time complexity for adaptive squeezing $\mathbf{q}, \mathbf{k}, \mathbf{v}$ is $\mathcal{O}(HW(2C_{qk}+C_v))$, the attention operation takes $\mathcal{O}((H^2+W^2)(C_{qk}+C_v))$ time and the adaptive expanding takes $\mathcal{O}(HW(2C_{qk}+C_v))$ time. Thus, our squeeze Axial attention successfully reduces time complexity to $\mathcal{O}(HW)$.

3.2.2 Squeeze Axial Position Embedding

Equation 4 are, however, not positional-aware, including no positional information of the feature map. Hence, we propose squeeze Axial position embedding to squeeze Axial attention. For squeeze Axial attention, we render both $\mathbf{q}_{(h)}$ and $\mathbf{k}_{(h)}$ to be aware of their position in the squeezed axial feature by introducing positional embedding $\mathbf{r}_{(h)}^q, \mathbf{r}_{(h)}^k \in \mathbb{R}^{H\times C_{qk}}$, which are linearly interpolated from learnable parameters $\mathbf{B}_{(h)}^q, \mathbf{B}_{(h)}^k \in \mathbb{R}^{L\times C_{qk}}$. L is a constant. In the same way, $\mathbf{r}_{(v)}^q, \mathbf{r}_{(v)}^k \in \mathbb{R}^{W\times C_{qk}}$ are applied to $\mathbf{q}_{(v)}, \mathbf{k}_{(v)}$. Thus, the positional-aware squeeze Axial attention can be expressed as Eq. 6.

$$\begin{aligned}\mathbf{y}_{(i,j)} = &\sum_{p=1}^{H} \text{softmax}_p\left((\mathbf{q}_{(h)i}+\mathbf{r}_{(h)i}^q)^\top(\mathbf{k}_{(h)p}+\mathbf{r}_{(h)p}^k)\right)\mathbf{v}_{(h)p} \\ &+ \sum_{p=1}^{W} \text{softmax}_p\left((\mathbf{q}_{(v)j}+\mathbf{r}_{(v)j}^q)^\top(\mathbf{k}_{(v)p}+\mathbf{r}_{(v)p}^k)\right)\mathbf{v}_{(v)p}\end{aligned} \tag{6}$$

3.2.3 Detail Enhancement Kernel

The squeeze operation, though extracting global semantic information efficiently, sacrifices the local details. Hence an auxiliary convolution-based kernel is applied to enhance the spatial details. As is shown in the upper path of Fig. 3, $\mathbf{q}, \mathbf{k}, \mathbf{v}$ are first get from $\mathbf{x}$ with another $\mathbf{W}_q^{(e)}, \mathbf{W}_k^{(e)} \in \mathbb{R}^{C_{qk}\times C}$, $\mathbf{W}_v^{(e)} \in \mathbb{R}^{C_v\times C}$ and are concatenated on the channel dimension and then passed to a block made up of 3×3 depth-wise convolution and batch normalization. By using a 3×3

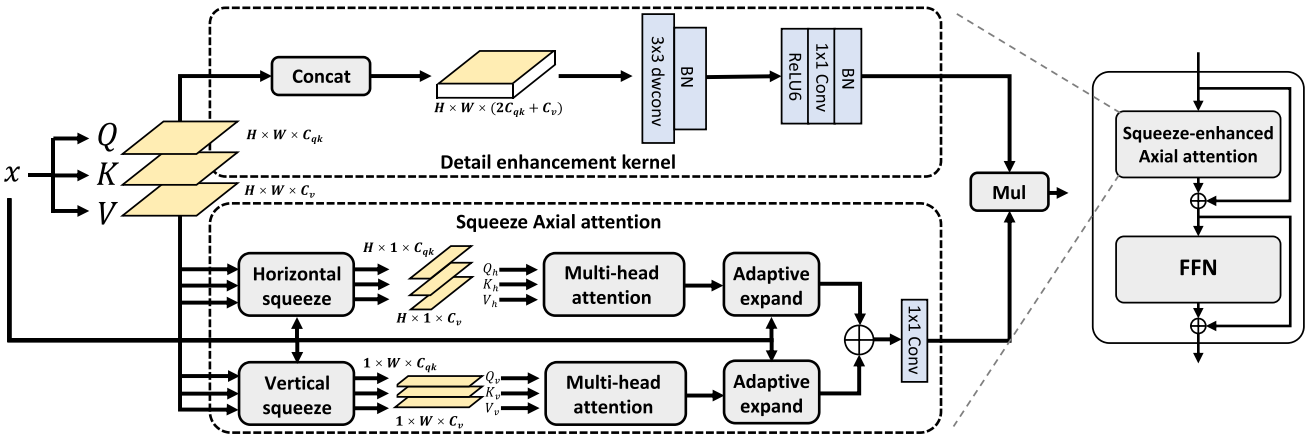


Fig. 3 ***Right***: the schematic illustration of the proposed squeeze-enhanced Axial Transformer layer including a squeeze-enhanced Axial attention and a Feed-Forward Network (FFN). ***Left*** is the squeeze-enhanced Axial Transformer layer, including detail enhancement kernel and squeeze Axial attention. The symbol $\bigoplus$ indicates an element-wise addition operation. Mul means multiplication

convolution, auxiliary local details can be aggregated from **q**, **k**, **v**. And then a linear projection with activation function and batch normalization is used to squeeze $(2C_{qk} + C_v)$ dimension to C and generate detail enhancement weights. Finally, the enhancement feature will be fused with the feature given by squeeze Axial attention. Different enhancement modes including element-wise addition and multiplication will be compared in the experiment section. Time complexity for the 3×3 depth-wise convolution is $\mathcal{O}(3^2HW(2C_{qk} + C_v))$ and the time complexity for the 1×1 convolution is $\mathcal{O}(HWC(2C_{qk} + C_v))$. Time for other operations like activation can be omitted.

3.2.4 Complexity Analysis

In our application, we set $C_{qk} = 0.5C_v$ to further reduce computation cost. The total time complexity of squeeze-enhanced Axial attention is

$$\begin{aligned}
&\mathcal{O}((H^2 + W^2)(C_{qk} + C_v) + \mathcal{O}(2HW(2C_{qk} + C_v)) \\
&\quad + \mathcal{O}((HWC + 9HW)(2C_{qk} + C_v)) \\
&= \mathcal{O}((1.5H^2 + 1.5W^2 + 4HW)C_v) \\
&\quad + \mathcal{O}((2HWC + 18HW)C_v) \\
&= \mathcal{O}(HW)
\end{aligned}$$

if we assume $H = W$ and take the channel as constant. SEA attention is linear to the feature map size theoretically. Moreover, SEA Attention only includes mobile-friendly operations like convolution, pooling, matrix multiplication and so on.

3.2.5 Architecture Details and Variants

SeaFormer backbone contains 6 stages, corresponding to the shared STEM and context branch in Fig. 2 in the main paper. When conducting the image classification experiments, a pooling layer and a linear layer are added at the end of the context branch.

Table 1 details the family of our SeaFormer configurations with varying capacities. We construct SeaFormer-Tiny, SeaFormer-Small, SeaFormer-Base and SeaFormer-Large models with different scales via varying the number of SeaFormer layers and the feature dimensions. We use input image size of 512 × 512 by default. For variants except SeaFormer-Large, SeaFormer layers are applied in the last two stages for a superior trade-off between accuracy and efficiency. For SeaFormer-Large, we apply the proposed SeaFormer layers in each stage of the context branch (Fig. 5).

3.3 Multi-resolution Distillation Based on Feature Up-Sampling

In dealing with dense prediction tasks, accurately extracting semantic information and spatial details from images often necessitates the input of high-resolution images. This approach undoubtedly increases the computational burden on the model, especially in resource-constrained environments such as mobile devices or real-time applications, where the demand for high computation becomes even more significant. In recent years, to alleviate this issue, some research (Qi et al., 2021; Hu et al., 2022) has proposed methods utilizing multi-scale distillation techniques. In these methods, a teacher model that takes high-resolution images as input is used to guide a student model that takes low-resolution

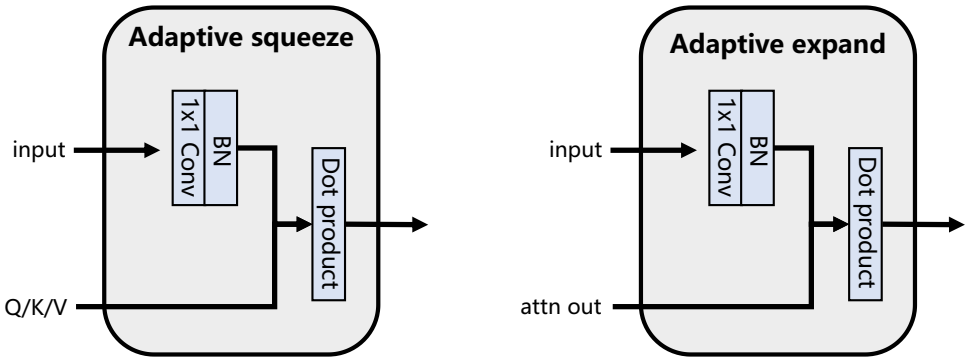


Fig. 4 ***Left***: the schematic diagram of the proposed adaptive squeezing. ***Right*** is the adaptive expanding operation. *Mat mul* means matrix multiplication. *Attn out* is the output of the multi-head attention

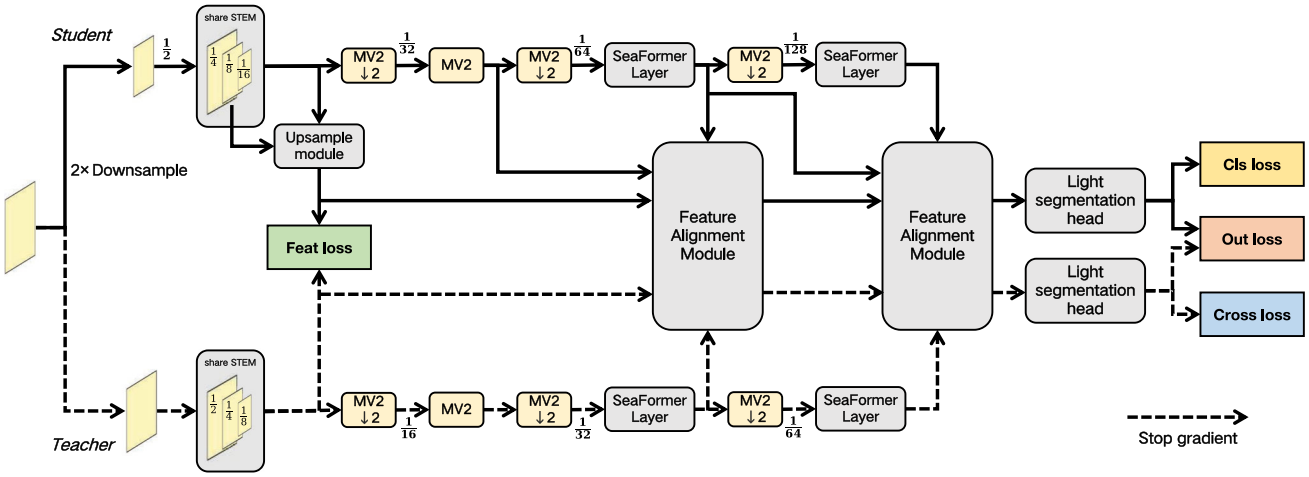


Fig. 5 The overall pipeline of multi-resolution distillation based on feature up-sampling. `MV2(E=4)` denotes MobileNetV2 block with an expansion ratio of 4, and the default kernel size for depth-wise convolution is 5

Table 1 Architectures for semantic segmentation

	Resolution	SeaFormer-Tiny	SeaFormer-Small	SeaFormer-Base	SeaFormer-Large
Stage1	H/2 × W/2	[Conv, 3, 16, 2]	[Conv, 3, 16, 2]	[Conv, 3, 16, 2]	[Conv, 3, 32, 2]
		[MB, 3, 1, 16, 1]	[MB, 3, 1, 16, 1]	[MB, 3, 1, 16, 1]	[MB, 3, 3, 32, 1]
Stage2	H/4 × W/4	[MB, 3, 4, 16, 2]	[MB, 3, 4, 24, 2]	[MB, 3, 4, 32, 2]	[MB, 3, 4, 64, 2]
		[MB, 3, 3, 16, 1]	[MB, 3, 3, 24, 1]	[MB, 3, 3, 32, 1]	[MB, 3, 4, 64, 1]
Stage3	H/8 × W/8	[MB, 5, 3, 32, 2]	[MB, 5, 3, 48, 2]	[MB, 5, 3, 64, 2]	[MB, 5, 4, 128, 2]
		[MB, 5, 3, 32, 1]	[MB, 5, 3, 48, 1]	[MB, 5, 3, 64, 1]	[MB, 5, 4, 128, 1]
Stage4	H/16 × W/16	[MB, 3, 3, 64, 2]	[MB, 3, 3, 96, 2]	[MB, 3, 3, 128, 2]	[MB, 3, 4, 192, 2]
		[MB, 3, 3, 64, 1]	[MB, 3, 3, 96, 1]	[MB, 3, 3, 128, 1]	[MB, 3, 4, 192, 1]
					[Sea, 3, 8]
Stage5	H/32 × W/32	[MB, 5, 3, 128, 2]	[MB, 5, 4, 160, 2]	[MB, 5, 4, 192, 2]	[MB, 5, 4, 256, 2]
		[Sea, 2, 4]	[Sea, 3, 6]	[Sea, 4, 8]	[Sea, 3, 8]
Stage6	H/64 × W/64	[MB, 3, 6, 160, 2]	[MB, 3, 6, 192, 2]	[MB, 3, 6, 256, 2]	[MB, 3, 6, 320, 2]
		[Sea, 2, 4]	[Sea, 3, 6]	[Sea, 4, 8]	[Sea, 3, 8]

[Conv, 3,16, 2] denotes regular convolution layer with kernel of 3, output channel of 16 and stride of 2. [MB, 3, 4, 16, 2] means MobileNetV2 (Sandler et al., 2018) block with kernel of 3, expansion ratio of 4, output channel of 16 and stride of 2. [Sea, 2, 4] refers to SeaFormer layers with number of layers of 2 and heads of 4

images as input to learn. This strategy allows the student model to produce reasonable results even with low-resolution inputs, thereby significantly reducing computational costs and increasing inference speed.

Inspired by the aforementioned methods, this paper proposes an innovative multi-resolution distillation approach based on feature upsampling. Unlike previous work, our method involves aligning the features of the teacher and student models at the same processing stage (even though the student model's input resolution is half that of the teacher model) and particularly emphasizes upsampling at the feature level to achieve alignment.

Specifically, assuming the student model's input resolution is half that of the teacher model, the resolution of the feature maps produced by the student model at any given forward propagation stage will naturally be half that of the corresponding stage in the teacher model. To address this mismatch, this study has designed a feature alignment module. This module employs a lightweight feature upsampling module constructed using MobileNetV2 to upsampling the student model's feature maps to the same resolution as that of the teacher model at the same stage. Subsequently, a feature similarity loss function is used to optimize the similarity between these two features, maximizing their consistency to better aid the student model in mimicking the behavior of the teacher model.

Furthermore, the upscaled features from the student model are not only used for alignment with the teacher model's features but are also integrated into the overall semantic segmentation framework, serving as spatial branch features in the feature fusion process. The specific method of this feature fusion and the involved modules are detailed in Sect. 3.1. Through this series of steps, the fused features are finally fed into a lightweight segmentation head to complete the semantic segmentation task.

3.4 Feature Alignment Module

Figure 6a shows the feature alignment module's goal to precisely align the student model's features with the teacher model's, maintaining architectural consistency as described in Sect. 3.1. This includes using feature fusion to merge spatial and current stage features, and employing the teacher's features as guides to improve the student's feature alignment and predictive accuracy. Initially, the student model's input resolution is halved via average pooling. To address this, a lightweight upsampling module, informed by the detailed matching features of both models, upsamples the student's features for alignment. Cosine similarity between the upscaled and teacher's features, with negative similarity contributing to the loss function, enhances the student's emulation of the teacher's features, ensuring efficient and accurate model performance despite lower input resolution.

3.4.1 Upsampling Module

Before alignment, the features of the student model are upsampled by applying a lightweight up-sampling module, which ensures that the feature resolution matches the feature resolution of the teacher model at the same stage, facilitating knowledge transfer and improving the performance of the student model. As demonstrated in Fig. 5, high-resolution feature maps from the previous stage are followed by a convolution, a batch normalization layer, and a sigmoid layer to produce weights to guide the up-sampling of low-resolution features in the current stage. Low-resolution feature maps are fed into a MobileNetV2 block and up-sampled to high resolution. Then, they are element-wise multiplied with each other, and the up-sampled features are refined with another MobileNetV2 block.

3.4.2 Loss Function

As indicated in Eq. 7, the loss function of the multi-resolution distillation method based on feature up-sampling comprises four components: classification loss, cross-model classification loss, feature similarity loss, and output similarity loss.

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{cross} + \mathcal{L}_{feat} + \mathcal{L}_{out} \tag{7}$$

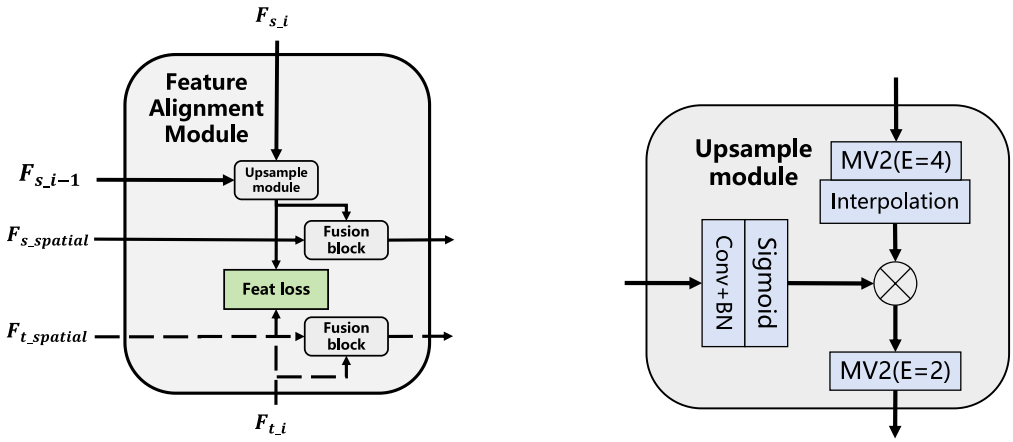
Classification loss $\mathcal{L}_{cls}$ refers to the cross-entropy loss between the output of the student model and the ground truth labels.

Cross-model classification loss $\mathcal{L}_{cross}$ denotes the cross-entropy loss between the output obtained by inputting the up-sampled features of the student model into the segmentation head of the teacher model and the ground truth labels.

The **feature similarity loss** $\mathcal{L}_{feat}$ measures the negative cosine similarity between the up-sampled features of the student model and the features of the teacher model at the corresponding stage.

The **output similarity loss** $\mathcal{L}_{out}$ represents the Kullback–Leibler Divergence between the output logits of the student model and the output logits of the teacher model.

The cross-model loss, feature similarity loss, and output similarity loss all contribute to the process of knowledge distillation. Although they differ in terms of the emphasis on the transmitted knowledge from the teacher model to the student model and the degree of relaxation in guiding model parameter updates, they all positively impact the learning of the student model. Our empirical study validates the effectiveness of the aforementioned loss functions.



(a) The schematic diagram of the feature alignment module.

(b) The schematic diagram of the upsample module.

Fig. 6 ***Left***: the schematic illustration of the proposed feature alignment module including an upsample module and a fusion block. F_{s_i} means feature map of student model on stage i. F_{t_i} means feature map of teacher model on stage i. $F_{s\_spatial}$ means feature map of student spatial branch. $F_{t\_spatial}$ means feature map of teacher spatial branch. ***Right*** is the upsample module

4 Experiments

We evaluate our method on semantic segmentation, image classification and object detection tasks. First, we describe implementation details and compare results with related efficient neural networks. We then conduct a series of ablation studies to validate the design of SeaFormer. Each proposed component and important hyper-parameters are examined thoroughly.

4.1 Experimental Setup

4.1.1 Dataset

We perform semantic segmentation experiments over ADE20K (Zhou et al., 2017), CityScapes (Cordts et al., 2016), Pascal Context (Mottaghi et al., 2014) and COCO-Stuff (Caesar et al., 2018). The mean of intersection over union (mIoU) is set as the evaluation metric. We convert full-precision models to TNN (Contributors, 2019) and measure latency on an ARM-based device with a single Qualcomm Snapdragon 865 processor.

ADE20K

ADE20K dataset covers 150 categories, containing 25K images that are split into 20K/2K/3K for *Train*, *val* and *test*.

CityScapes

CityScapes is a driving dataset for semantic segmentation. It consists of 5000 fine annotated high-resolution images with 19 categories.

PASCAL Context

dataset has 4998 scene images for training and 5105 images for testing. There are 59 semantic labels and 1 background label.

COCO-Stuff

dataset augments COCO dataset with pixel-level stuff annotations. There are 10000 complex images selected from COCO. The training set and test set consist of 9K and 1K images respectively.

4.1.2 Implementation Details

We set ImageNet-1K (Deng et al., 2009) pretrained network as the backbone, and training details of ImageNet-1K are illustrated in the last subsection. For semantic segmentation, the standard BatchNorm (Ioffe & Szegedy, 2015) layer is replaced by synchronized BatchNorm. Our implementation is based on public codebase `mmsegmentation` (Contributors, 2020). We follow the batch size, training iteration scheduler and data augmentation strategy of TopFormer (Zhang et al., 2022) for a fair comparison.

ADE20K

The initial learning rate is 0.0005 for batch size 32 or 0.00025 for batch size 16. The weight decay is 0.01. A poly learning rate scheduled with factor 1.0 is adopted.

Cityscapes

The initial learning rate is 0.0003 and the weight decay is 0.01. The comparison of Cityscapes contains full-resolution and half-resolution. For the full-resolution version, the training images are randomly scaled and then cropped to the fixed size of 1024×1024. For the half-resolution version, the training images are resized to 1024×512 and randomly scaling, the crop size is 1024×512.

Pascal Context

The initial learning rate is 0.0002 and the weight decay is 0.01. A poly learning rare scheduled with factor 1.0 is used.

Table 2 Results of semantic segmentation on ADE20K *val* set, * indicates training batch size is 32

Backbone	Decoder	Params	FLOPs	mIoU	Latency
MobileNetV2	LR-ASPP	2.2M	2.8G	32.0	177ms
MobileNetV3-Lr	LR-ASPP	1.6M	1.3G	32.3	81ms
MobileNetV3-Large	LR-ASPP	3.2M	2.0G	33.1	126ms
HRNet-W18-Small	HRNetW18S	4.0M	10.2G	33.4	639ms
TopFormer-T	Simple Head	1.4M	0.6G	33.6	43ms
TopFormer-T*	Simple Head	1.4M	0.6G	34.6	43ms
PP-MobileSeg-T*	Seg Head	1.5M	0.7G	36.4	47ms
SeaFormerT	Light Head	1.7M	0.6G	35.0	40ms
SeaFormer-T*	Light Head	1.7M	0.6G	35.8	40ms
SeaFormer-T++	Light Head	1.8M	0.6G	**36.8**	41ms
SeaFormer-T++(KD)	Light Head	2.3M	0.3G	35.5	**22ms**
ConvMLP-S	SemanticFPN	12.8M	33.8G	35.8	777ms
EfficientNet	DeepLabV3+	17.1M	26.9G	36.2	970ms
MobileNetV2	Lite-ASPP	2.9M	4.4G	36.6	235ms
TopFormer-S	Simple Head	3.1M	1.2G	36.5	74ms
TopFormer-S*	Simple Head	3.1M	1.2G	37.0	74ms
SeaFormer-S	Light Head	4.0M	1.1G	38.1	67ms
SeaFormer-S*	Light Head	4.0M	1.1G	39.4	67ms
SeaFormer-S++	Light Head	4.1M	1.1G	**39.7**	68ms
SeaFormer-S++(KD)	Light Head	5.0M	0.5G	38.4	**33ms**
MiT-B0	SegFormer	3.8M	8.4G	37.4	770ms
ResNet18	Lite-ASPP	12.5M	19.2G	37.5	648ms
ShuffleNetV2-1.5x	DeepLabV3+	16.9M	15.3G	37.6	960ms
MobileNetV2	DeepLabV3+	15.4M	25.8G	38.1	1035ms
TopFormer-B	Simple Head	5.1M	1.8G	38.3	110ms
TopFormer-B*	Simple Head	5.1M	1.8G	39.2	110ms
SeaFormer-B	Light Head	8.6M	1.8G	40.2	106ms
SeaFormer-B*	Light Head	8.6M	1.8G	41.0	106ms
SeaFormer-B++	Light Head	8.7M	1.8G	**41.4**	107ms
SeaFormer-B++(KD)	Light Head	10.2M	0.9G	39.5	**55ms**
MiT-B1	SegFormer	13.7M	15.9G	41.6	1300ms
SeaFormer-L	Light Head	14.0M	6.5G	42.7	367ms
SeaFormer-L*	Light Head	14.0M	6.5G	43.7	367ms
SeaFormer-L++	Light Head	14.1M	6.5G	**43.8**	369ms
SeaFormer-L++(KD)	Light Head	17.1M	2.9G	42.2	**177ms**

KD means knowledge distillation. The latency is measured on a single Qualcomm Snapdragon 865 with input size 512×512, and only an ARM CPU core is used for speed testing. MobileNetV3-Lr means MobileNetV3-Large-reduce. HRNet-W18S means HRNet-W18-Small. References: MobileNetV2 (Sandler et al., 2018), MobileNetV3 (Howard et al., 2019), HRNet (Yuan et al., 2020), TopFormer (Zhang et al., 2022), PP-MobileSeg (Tang et al., 2023), ConvMLP (Li et al., 2023), Semantic FPN (Kirillov et al., 2019), EfficientNet (Tan & Le, 2019), DeepLabV3+ and Lite-ASPP (Chen et al., 2018), SegFormer (Xie et al., 2021), ResNet (He et al., 2016), ShuffleNetV2-1.5x (Ma et al., 2018)

COCO-Stuff

The initial learning rate is 0.0002 and the weight decay is 0.01. A poly learning rare scheduled with factor 1.0 is used.

During inference, we set the same resize and crop rules as TopFormer to ensure fairness.

4.2 Comparison with State of the Art

ADE20K

Table 2 shows the results of SeaFormer and previous efficient backbones on ADE20K *val* set. The comparison covers Params (model parameters), FLOPs (floating point operations), Latency and mIoU. As shown in Table 2, SeaFormer

Table 3 Results on Cityscapes *val* and *test* set

Method	Backbone	FLOPs	mIoU(val)	mIoU(test)	Latency
FCN	MobileNetV2	317 G	61.5	–	24190ms
PSPNet	MobileNetV2	423 G	70.2	–	31440ms
SegFormer(h)	MiT-B0	17.7G	71.9	-	1586ms
SegFormer(f)	MiT-B0	125.5G	76.2	–	11030ms
L-ASPP	MobileNetV2	12.6G	72.7	–	887ms
LR-ASPP	MobileNetV3-L	9.7G	72.4	72.6	660ms
LR-ASPP	MobileNetV3-S	2.9G	68.4	69.4	211ms
SimpleHead(h)	TopFormer-B	2.7G	70.7	–	173ms
SimpleHead(f)	TopFormer-B	11.2G	75.0	75.0	749ms
Light Head(h)	**SeaFormerS**	2.0G	70.7	71.0	129ms
Light Head(h)	**SeaFormerS++**	2.0G	72.1	72.3	130ms
Light Head(f)	**SeaFormerS**	8.0G	76.1	75.9	518ms
Light Head(f)	**SeaFormerS++**	8.0G	77.2	76.9	521ms
Light Head(h)	**SeaFormerB**	3.4G	72.2	72.5	205ms
Light Head(h)	**SeaFormerB++**	3.4G	73.5	73.4	207ms
Light Head(f)	**SeaFormerB**	13.7G	77.7	77.5	821ms
Light Head(f)	**SeaFormerB++**	13.7G	78.6	78.3	825ms

The results on *test* set of some methods are not presented due to the fact that they are not reported in their original papers

outperforms these efficient approaches with comparable or less FLOPs and lower latency. Compared with specially designed mobile backbone, TopFormer, which sets global self-attention as its semantics extractor, SeaFormer achieves higher segmentation accuracy with lower latency. And the performance of SeaFormer-B++ surpasses MobileNetV3 by a large margin of +8.3% mIoU with lower latency (-16%). The results demonstrate that our SeaFormer layers improve the representation capability significantly.

Cityscapes

From the Table 3, it can be seen that SeaFormer-B++ is 1.3 points better than SeaFormer-B with only a slight increase in latency, showing the benefit of our efficient architecture design with multiple SeaFormer layers embedded in. It is worth noting that with less computation cost and latency, our SeaFormer-S and SeaFormer-S++ even outperform TopFormer-B. This result further confirms the performance and efficiency of our model when processing high-resolution input images.

Pascal Context

We compare SeaFormer with the previous approaches on Pascal Context validation set in Table 4. We evaluate the performance over 59 categories and 60 categories (including background). From the results, it can be seen that SeaFormer-S++ is +1.2% mIoU higher (46.31% vs.45.08%) than SeaFormer-S with similar latency.

COCO-Stuff

We compare SeaFormer with the previous approaches on COCO-Stuff validation set in Table 5. From the results, it can be seen that SeaFormer-S++ is +1.2% mIoU higher (34.04 *vs.*32.82) than SeaFormer-S with a similar computation cost.

Table 4 Results on Pascal Context *val* set

Backbone	Decoder	FLOPs	mIoU (60/59)
MBV2-s16	DeepLabV3+	22.24G	38.59/42.34
ENet-s16	DeepLabV3+	23.00G	39.19/43.07
MBV3-s16	LR-ASPP	2.04G	35.05/38.02
TopFormer-T	Simple Head	0.53G	36.41/40.39
SeaFormer-T	Light Head	0.51G	37.27/41.49
SeaFormer-T++	Light Head	0.52G	**38.61/42.56**
TopFormer-S	Simple Head	0.98G	39.06/43.68
SeaFormer-S	Light Head	0.98G	40.20/45.08
SeaFormer-S++	Light Head	1.00G	**41.44/46.31**
TopFormer-B	Simple Head	1.54G	41.01/45.28
SeaFormer-B	Light Head	1.57G	41.77/45.92
SeaFormer-B++	Light Head	1.60G	**42.52/46.40**

We omit the latency as the input resolution is almost the same as that in Table 1

4.3 Ablation Studies

In this section, we ablate different self-attention implementations and some important design elements in the proposed model, including our squeeze-enhanced Axial attention module (SEA attention) and fusion block on ADE20K dataset.

Table 5 Results on COCO-Stuff *test* set

Backbone	Decoder	FLOPs	mIoU
MBV2-s8	PSPNet	52.94G	30.14
ENet-s16	DeepLabV3+	27.10G	31.45
MBV3-s16	LR-ASPP	2.37G	25.16
TopFormer-T	Simple Head	0.64G	28.34
SeaFormer-T	Light Head	0.62G	29.24
SeaFormer-T++	Light Head	0.63	**30.76**
TopFormer-S	Simple Head	1.18G	30.83
SeaFormer-S	Light Head	1.15G	32.82
SeaFormer-S++	Light Head	1.17G	**34.04**
TopFormer-B	Simple Head	1.83G	33.43
SeaFormer-B	Light Head	1.81G	34.07
SeaFormer-B++	Light Head	1.84G	**35.01**

We omit the latency in this table as the input resolution is the same as that in Table 1

The influence of components in SEA attention

We conduct experiments with several configurations, including detail enhancement kernel only, squeeze Axial attention only, and the fusion of both. As is shown in Table 6, only detail enhancement or squeeze Axial attention achieves a relatively poor performance, and enhancing squeeze Axial attention with detail enhancement kernel brings a performance boost with a gain of 2.3% mIoU on ADE20K. The results indicate that enhancing global semantic features from squeeze Axial attention with local details from convolution optimizes the feature extraction capability of Transformer block. For enhancement input, there is an apparent performance gap between upconv(x) and conv(x). And we conclude that increasing the channels will boost performance significantly. Comparing concat[qkv] and upconv(x), which also correspond to w/ or w/o convolution weight sharing between detail enhancement kernel and squeeze Axial attention, we can find that sharing weights makes our model improve inference efficiency with minimal performance loss (35.8 *vs.*35.9). As for enhancement modes, multiplying features from squeeze Axial attention and detail enhancement kernel outperforms add enhancement by +0.4% mIoU (Table 7).

Comparing Different Self-attention Modules in the Swin Transformer

To eliminate the impact of our architecture and demonstrate the effectiveness and generalization ability of SEA attention, we ran experiments on Swin Transformer (Liu et al., 2021) by replacing window attention in Swin Transformer with different attention blocks. We set the same training protocol, hyper-parameters, and model architecture configurations as Swin for a fair comparison. When replacing window attention with CCAttention (CCNet) or DoubleAttention (A2-Nets), they have much lower FLOPs than SeaFormer and other attention blocks. Considering that we may not be able to draw conclusions rigorously, we doubled the number of their Transformer blocks (including MLP). As ACmix has the same architecture configuration as Swin, we borrow the results from the original paper. From Table 9, it can be seen that SeaFormer outperforms other attention mechanisms with lower FLOPs and latency.

Comparing Different Self-attention modules in the SeaFormer

To verify the effectiveness and efficiency of SEA attention based on our designed pipeline, we experiment with convolution, Global attention, Local attention, Axial attention and three convolution enhanced attention methods including our SEA attention, ACmix and MixFormer. The ablation experiments are organized in seven groups. Since the resolution of computing attention is relatively small, the window size in Local attention, ACmix, and MixFormer is set to 4. We adjust the channels when applying different attention modules to

Table 6 Ablation studies on components in SEA attention on ImageNet-1K and ADE20K datasets

Enhance kernel	Attn branch	Enhance input	Enhance mode	Params	FLOPs	Latency	Top1	mIoU
✔		–	–	1.3M	0.58G	38ms	65.9	32.5
	✔	–	–	1.4M	0.57G	38ms	66.3	33.5
✔	✔	conv(x)	Mul	1.6M	0.60G	40ms	67.2	34.9
✔	✔	upconv(x)	Mul	1.8M	0.62G	41ms	68.1	35.9
✔	✔	concat[qkv]	Mul	1.7M	0.60G	40ms	67.9	35.8
✔	✔	concat[qkv]	Add	1.7M	0.60G	40ms	67.3	35.4

Enhancement input means the input of detail enhancement kernel. conv(x) means x followed by a point-wise conv. upconv(x) is the same as conv(x) except different channels as upconv(x) is from C_{in} to $C_q + C_k + C_v$ and conv(x) is from C_{in} to C_{in}. concat[qkv] indicates concatenation of Q,K,V

Table 7 Ablation studies on squeeze and expand methods in SEA attention on ImageNet-1K and ADE20K datasets

Squeeze method	Expand method	Params	FLOPs	Latency	Top1	mIoU
Mean pooling	Broadcast	1.7M	0.60G	40ms	67.9	35.8
Max pooling	Broadcast	1.7M	0.60G	40ms	67.4	35.0
Adaptive squeeze	Adaptive expand	1.8M	0.61G	41ms	69.8	36.8

Table 8 Comparison of different architecture with our proposed SEA attention

Architecture	Params	FLOPs	Latency	mIoU
DeepLabV3+ (Chen et al., 2018)	17.1M	24.5G	710ms	39.3
SegFormer (Xie et al., 2021)	17.1M	15.2G	920ms	42.7
Swin (Liu et al., 2021)	34.0M	24.9G	2278ms	46.5
SeaFormer++	14.1M	6.5G	369ms	43.8
SeaFormer++ (KD)	17.1M	2.9G	177ms	42.2

Table 9 Results on ADE20K *val* set based on Swin Transformer architecture

Model	Params(B)	FLOPs(B)	mIoU	Latency
Swin	27.5M	25.6G	44.5	3182ms
CCNet	41.6M	37.4G	43.1	3460ms
ISSA	31.8M	33.3G	37.4	2991ms
A2-Nets	37.2M	31.1G	28.9	2502ms
Axial	36.2M	32.5G	45.3	3121ms
Local	27.5M	25.1G	34.2	3059ms
MixFormer	27.5M	24.9G	45.5	2817ms
ACmix	27.9M	26.6G	45.3	3712ms
Global	27.5M	0.144T	OOM	14642ms
SeaFormer	34.0M	**24.9G**	**46.5**	**2278ms**

(B) denotes backbone. OOM means CUDA out of memory. References: ISSA (Huang et al., 2019), A2-Nets (Chen et al., 2018)

Table 10 Performance of different self-attention modules on our designed pipeline on ImageNet-1K and ADE20K datasets

Method	Params	FLOPs	Latency	Top1	mIoU
Conv	1.6M	0.59G	38ms	66.3	32.8
Local	1.3M	0.60G	48ms	65.9	32.8
Axial	1.6M	0.63G	44ms	66.9	33.7
Global	1.3M	0.61G	43ms	66.7	34.2
ACmix	1.3M	0.60G	54ms	66.0	33.1
MixFormer	1.3M	0.60G	50ms	66.8	33.8
SeaFormer	1.7M	0.60G	**40ms**	**67.9**	**35.8**

Table 11 Ablation study on fusion method on ADE20K *val* set

Fusion method	mIoU
Add directly	35.2
Multiply directly	35.2
Sigmoid add	34.8
Sigmoid multiply	**35.8**

keep the FLOPs aligned and compare their performance and latency. The results are illustrated in Table 10.

As demonstrated in the table, SEA attention outperforms the counterpart built on other efficient attentions. Compared with global attention, SEA attention outperforms it by +1.2% Top1 accuracy on ImageNet-1K and +1.6 mIoU on ADE20K with less FLOPs and lower latency. Compared with similar convolution enhanced attention works, ACmix and MixFormer, our SEA attention obtains better results on ImageNet-1K and ADE20K with similar FLOPs but lower latency. The results indicate the effectiveness and efficiency of SEA attention module.

Comparing Different Architecture with the Same Attention Module

Table 12 Ablation studies on embedding dimensions and position bias

Embed dim	Params	FLOPs	Latency	mIoU
64,96	8.5M	1.7G	102ms	40.3
128,160	8.6M	1.8G	106ms	41.0
192,256	8.7M	2.0G	121ms	41.2
Position bias	**Params**	**FLOPs**	**Latency**	**mIoU**
✘	1.65M	0.60G	40ms	35.6
✔	1.67M	0.60G	40ms	35.8

[128, 160] is an optimal embedding dimension in fusion blocks

Table 8 provides a comprehensive comparison of different architectures with the same attention module. Our proposed architecture, SeaFormer++, demonstrates a favorable balance between efficiency and accuracy. Notably, SeaFormer++ achieves a latency of 369ms with only 6.5G FLOPs, which is significantly lower than Swin Transformer (2278ms and 24.9G FLOPs) while maintaining competitive mIoU performance (43.8 vs. 46.5).

Additionally, with the inclusion of knowledge distillation (SeaFormer++ KD), the model's latency further improves to an impressive 177ms with only 2.9G FLOPs, while still maintaining a mIoU of 42.2. This demonstrates the impact of knowledge distillation in not only reducing computational cost but also preserving performance. Compared to DeepLabV3+ and SegFormer, our SeaFormer++ with KD achieves comparable accuracy (42.2 mIoU vs. 42.7 mIoU for SegFormer) while offering much faster inference times and fewer FLOPs, confirming the efficiency of the overall architecture design and distillation pipeline.

The Influence of Squeeze and Expand Method

To evaluate different squeeze and expand strategies within the SEA attention framework, we conducted a structured series of ablation studies. These were divided into three groups based on the method used, focusing on maintaining consistent FLOPs, latency, and comparative performance on the ImageNet-1K and ADE20K datasets.

Table 7 summarizes the outcomes. Notably, the 'Adaptive squeeze and Adaptive expand' method excelled, achieving 69.8% Top1 accuracy on ImageNet-1K and 36.8 mIoU on ADE20K. The 1x1 convolutional layers in the adaptive squeeze and expand modules contribute a minor increase in parameter count (+0.1M) and computational cost (+0.01G), which have a negligible impact on latency (+1ms). However, these modifications result in significant performance improvement, demonstrating the effectiveness of our design choices.

The Influence of Fusion Block Design

We set four fusion methods, including Add directly, Multiply directly, Sigmoid add and Sigmoid multiply. $\mathbf{X}$ directly means features from context branch and spatial branch $\mathbf{X}$

Table 13 Comparison of different heads and backbones across various models

Model	Head	Params	FLOPs	Latency	mIoU
Topformer-Tiny*	Simple head	1.4M	0.58G	43ms	34.6
Topformer-Tiny*	Light head	1.4M	0.64G	47ms	33.1
SeaFormer-Tiny*	Simple head	1.7M	0.60G	42ms	35.6
SeaFormer-Tiny*	Light head	1.7M	0.60G	**40ms**	35.8
SeaFormer-Tiny++	Light head	1.8M	0.61G	41ms	**36.8**
Topformer-Small*	Simple head	3.1M	1.16G	74ms	37.0
Topformer-Small*	Light head	3.1M	1.19G	76ms	35.3
SeaFormer-Small*	Simple head	4.0M	1.10G	70ms	38.8
SeaFormer-Small*	Light head	4.0M	1.10G	**67ms**	39.4
SeaFormer-Small++	Light head	4.1M	1.11G	68ms	**39.7**
Topformer-Base*	Simple head	5.1M	1.81G	110ms	39.2
Topformer-Base*	Light head	5.1M	1.85G	112ms	36.6
SeaFormer-Base*	Simple head	8.7M	1.84G	108ms	40.6
SeaFormer-Base*	Light head	8.6M	1.82G	**106ms**	41.0
SeaFormer-Base++	Light head	8.7M	1.83G	107ms	**41.4**

* indicates training batch size is 32

directly. Sigmoid **X** means feature from context branch goes through a sigmoid layer and **X** feature from spatial branch.

From Table 11 we can see that replacing sigmoid multiply with other fusion methods hurts performance. Sigmoid multiply is our optimal fusion block choice.

The Influence of the Width in Fusion Block

To study the influence of the width in fusion block, we perform experiments with different embedding dimensions in fusion blocks on SeaFormer-Base, M denotes the channels that spatial branch and context branch features mapping to in two fusion blocks. Results are shown in Table 12.

Additional Ablation Studies

To justify the effectiveness of the backbone and decoder in SeaFormer++, we have conducted additional experiments comparing different combinations of backbones and heads. The results are shown in Table 13.

We used the same backbone with different heads, including the Simple head from the most related baseline Topformer (Zhang et al., 2022) and the Light head proposed in our work. Compared to the Simple head in SeaFormer-Base, our proposed Light head gains 0.4 mIoU improvement with a decrease of 0.1M parameters, 0.02G FLOPs, and 2ms Latency. Additionally, we compared different backbones (Topformer, SeaFormer, and SeaFormer++) using the same proposed Light head, SeaFormer-Base++ outperforms TopFormer-Base (41.4 *vs.*36.6) with lower latency (107ms *vs.*112ms).

The results in the table highlight the superior performance of the SeaFormer++ backbone and Light head in terms of both efficiency (latency and FLOPs) and effectiveness (mIoU). SeaFormer++ consistently outperforms Topformer

Table 14 Image classification results on ImageNet-1K *val* set

Method	Params	FLOPs	Top1	Latency
MobileNetV3-Small (Howard et al., 2019)	2.9M	0.1G	67.4	**5ms**
SeaFormer-T	1.8M	0.1G	67.9	7ms
SeaFormer-T++	1.9M	0.1G	**69.8**	7ms
MobileViT-XXS (Mehta and Rastegari, 2022)	1.3M	0.4G	69.0	24ms
MobileViTv2−0.50 (Mehta and Rastegari, 2022)	1.4M	0.5G	70.2	32ms
MobileOne-S0* (Vasu et al., 2023)	2.1M	0.3G	71.4	14ms
MobileNetV2 (Sandler et al., 2018)	3.4M	0.3G	72.0	17ms
Mobile-Former96 (Chen et al., 2022)	4.8M	0.1G	72.8	31ms
SeaFormer-S	4.1M	0.2G	73.3	12ms
SeaFormer-S++	4.2M	0.2G	**74.5**	**12ms**
EdgeViT-XXS (Pan et al., 2022)	4.1M	0.6G	74.4	71ms
LVT (Yang et al., 2022)	5.5M	0.9G	74.8	97ms
MobileViT-XS (Mehta and Rastegari, 2022)	2.3M	0.9G	74.8	54ms
MobileNetV3-Large (Howard et al., 2019)	5.4M	0.2G	75.2	**16ms**
Mobile-Former151 (Chen et al., 2022)	7.7M	0.2G	75.2	42ms
MobileViTv2−0.75 (Mehta and Rastegari, 2022)	2.9M	1.0G	75.6	68ms
EfficientFormerV2-S0 (Li et al., 2023)	3.5M	0.4G	75.7	25ms
MobileOne-S1* (Vasu et al., 2023)	4.8M	0.8G	75.9	40ms
SeaFormer-B	8.7M	0.3G	76.0	20ms
SeaFormer-B++	8.8M	0.3G	**77.0**	20ms
MobileOne-S2* (Vasu et al., 2023)	7.8M	1.3G	77.4	63ms
EdgeViT-XS (Pan et al., 2022)	6.8M	1.1G	77.5	124ms
MobileViTv2−1.00 (Mehta and Rastegari, 2022)	4.9M	1.8G	78.1	115ms
MobileOne-S3* (Vasu et al., 2023)	10.1M	1.9G	78.1	91ms
AxialFormer (Wang et al., 2020)	11.6M	3.0G	78.1	151ms
MobileViT-S (Mehta and Rastegari, 2022)	5.6M	1.8G	78.4	88ms
EfficientNet-B1 (Tan & Le, 2019)	7.8M	0.7G	79.1	61ms
EfficientFormer-L1 (Li et al., 2022)	12.3M	1.3G	79.2	94ms
Mobile-Former508 (Chen et al., 2022)	14.8M	0.5G	79.3	102ms
MobileOne-S4* (Vasu et al., 2023)	14.8M	3.0G	79.4	143ms
SeaFormer-L	14.0M	1.2G	79.9	61ms
SeaFormer-L++	14.1M	1.2G	**80.6**	**61**ms

The FLOPs and latency are measured with input size 224×224, except for MobileViT and MobileViTv2 which are measured with 256×256 according to their original implementations. * indicates re-parameterized variants (Vasu et al., 2023). The latency is measured on a single Qualcomm Snapdragon 865, and only an ARM CPU core is used for speed testing. No other means of acceleration, e.g., GPU or quantification, is used

Table 15 Results on COCO object detection

Backbone	AP	FLOPs	Params
ShuffleNetv2 (Ma et al., 2018)	25.9	161 G	10.4M
SeaFormer-T	31.5	160 G	10.9M
SeaFormer-T++	**32.8**	**160 G**	11.0M
Mobile-Former151 (Chen et al., 2022)	34.2	161 G	14.4M
MobileNetV3 (Howard et al., 2019)	27.2	162 G	12.3M
SeaFormer-S	34.6	161 G	13.3M
SeaFormer-S++	**35.5**	**161 G**	13.4M
Mobile-Former214 (Chen et al., 2022)	35.8	**162 G**	15.2M
Mobile-Former294 (Chen et al., 2022)	36.6	164 G	16.1M
SeaFormer-B	36.7	164 G	18.1M
SeaFormer-B++	**37.4**	164 G	18.2M
ResNet50 (He et al., 2016)	36.5	239 G	37.7M
PVT-Tiny (Wang et al., 2021)	36.7	221 G	23.0M
AxialFormer (Wang et al., 2020)	37.7	210 G	23.7M
ConT-M (Yan et al., 2021)	37.9	217 G	27.0M
SeaFormer-L	39.8	185 G	24.0M
SeaFormer-L++	**40.2**	**185 G**	24.1M

across various configurations, confirming the strength of our backbone and head design.

4.4 Image Classification

We conduct experiments on ImageNet-1K (Deng et al., 2009), which contains 1.28M training images and 50K validation images from 1,000 classes. We employ an AdamW (Kingma, 2014) optimizer for 600 epochs using a cosine decay learning rate scheduler. A batch size of 1024, an initial learning rate of 0.064, and a weight decay of 2e-5 are used. The results are illustrated in Table 14. Compared with other efficient approaches, SeaFormer++ achieves a relatively better trade-off between latency and accuracy.

5 Object Detection

To further demonstrate the generalization ability of our proposed SeaFormer++ backbone on other downstream tasks, we conduct object detection task on COCO dataset.

We use RetinaNet (Lin, 2017) (one-stage) as the detection framework and follow the standard settings to use SeaFormer++ as backbone to generate e feature pyramid at multiple scales. All models are trained on train2017 split for 12 epochs (1×) from ImageNet pretrained weights.

From the Table 15, we can observe that our SeaFormer++ achieves superior results on detection task which further demonstrates the strong generalization ability of our method.

Table 16 Performance comparison on ADE20K *val* set under different precision

Model	mIoU	FP32	FP16
TopFormer-T	34.6	43ms	23ms
SeaFormer-T	35.8	40ms	22ms
SeaFormer-T++	**36.8**	**41ms**	**23ms**
TopFormer-S	37.0	74ms	41ms
SeaFormer-S	39.4	67ms	36ms
SeaFormer-S++	**39.7**	**68ms**	**37ms**
TopFormer-B	39.2	110ms	60ms
SeaFormer-B	41.0	106ms	56ms
SeaFormer-B++	**41.4**	**107ms**	**56ms**
SeaFormer-L	43.7	367ms	186ms
SeaFormer-L++	**43.8**	**369ms**	**187ms**

5.1 Latency statistics

We make the statistics of the latency of the proposed SeaFormer-Tiny, as shown in Fig. 10, the shared STEM takes up about half of the latency of the whole network (49%). The latency of the context branch is about a third of the total latency (34%), whilst the actual latency of the spatial branch is relatively low (8%) due to sharing early convolution layers with the context branch. Our light segmentation head (8%) also contributes to the success of building a light model.

6 Performance Under Different Precision of the Models

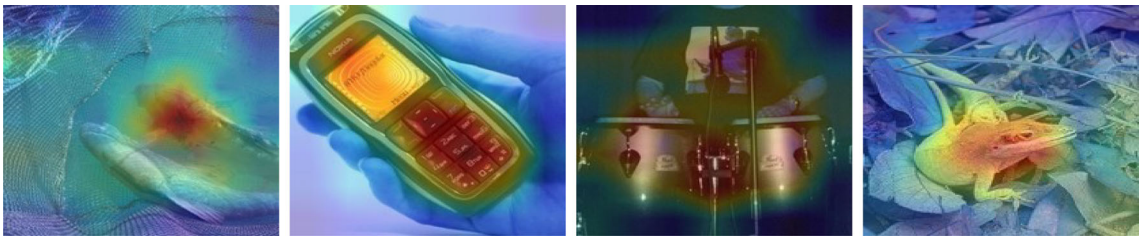
Following TopFormer, we measure the latency in the submission paper on a single Qualcomm Snapdragon 865, and only an ARM CPU core is used for speed testing. No other means of acceleration, e.g., GPU or quantification, is used. We provide a more comprehensive comparison to demonstrate the necessity of our proposed method. We test the latency under different precision of the models. From the Table 16, it can be seen that whether it is full precision or half precision the performance of SeaFormer is better than that of TopFormer.

7 Visualization

7.1 Attention Heatmap

To demonstrate the effectiveness of detail enhancement in our squeeze-enhanced Axial attention (SEA attention), we ablate our model by removing the detail enhancement. We visualize the attention heatmaps of the two models in Fig. 7. Without detail enhancement, attention heatmaps from solely

(a) Squeeze Axial attention heatmaps



(b) Squeeze-enhanced Axial attention heatmaps

Fig. 7 The visualization of attention heatmaps from the model consisting of squeeze axial attention without detail enhancement (*first row*) and SeaFormer (*second row*). Heatmaps are produced by averaging channels of the features from the last attention block, normalizing to [0, 255], and up-sampling to the image size

(a) Ground Truth

(b) TopFormer-B [32]

(c) SeaFormer-B (Ours)

Fig. 8 Visualization of prediction results on ADE20K *val* set

(a) Ground Truth

(b) SegFormer-B1 [4]

(c) SeaFormer-L (Ours)

Fig. 9 Visualization of prediction results on ADE20K *val* set

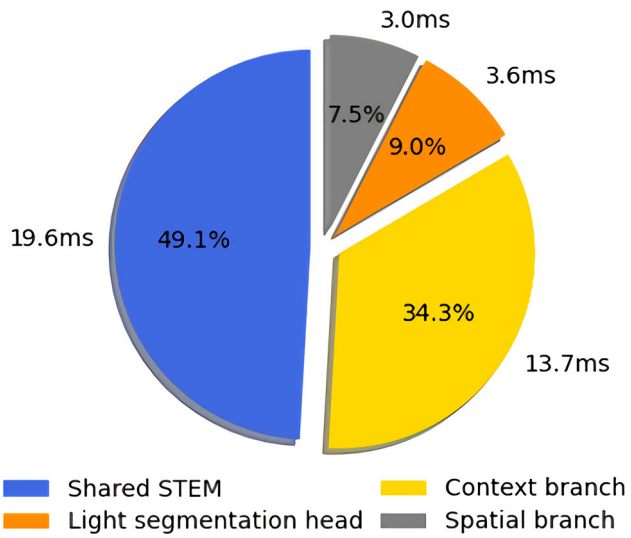


Fig. 10 The inference latency of components

SA attention appear to be axial strips while our proposed SEA attention is able to activate the semantic local region accurately, which is particularly significant in the dense prediction task.

7.2 Prediction Results

We show the qualitative results and compare with the alternatives on the ADE20K validation set from two different perspectives. First, we compare with a mobile-friendly rival TopFormer (Zhang et al., 2022) with similar FLOPs and latency in Fig. 8. Besides, we compare with the Transformer-based counterpart SegFormer-B1 (Xie et al., 2021) in Fig. 9. In particular, our SeaFormer-L has a lower computation cost than the SegFormer-B1. As shown in both figures, we demonstrate better segmentation results than both the mobile counterpart and Transformer-based approach.

8 Multi-resolution Distillation Based on Feature Up-Sampling

8.1 Experimental Setup

We perform multi-resolution distillation experiments over ADE20K (Zhou et al., 2017). mIoU is set as the evaluation metric. We convert full-precision models to TNN (Contributors, 2019) and measure latency on an ARM-based device with a single Qualcomm Snapdragon 865 processor. We set ADE20K fine-tuned network with original resolution input in Sect. 4 as the teacher model and ImageNet-1K (Deng et al., 2009) pretrained network as the backbone of the student model. The input resolution of the student model is halved by average pooling.

The standard BatchNorm (Ioffe & Szegedy, 2015) layer is replaced by synchronized BatchNorm. The implementation of multi-resolution distillation is based on public codebase `mmsegmentation` (Contributors, 2020). We follow the batch size, training iteration scheduler and data augmentation strategy of TopFormer (Zhang et al., 2022) and Sect. 4 for a fair comparison. The initial learning rate is 0.0005 for batch size 32 or 0.00025 for batch size 16. The weight decay is 0.01. A poly learning rate scheduled with factor 1.0 is adopted. During inference, we set the same resize and crop rules as TopFormer to ensure fairness.

8.2 Comparison with State of the Art

Table 2 shows the results of SeaFormer++ (KD) and previous efficient backbones on ADE20K *val* set. The comparison covers Params, FLOPs, Latency and mIoU. As shown in Table 2, SeaFormer++ (KD) outperforms these efficient approaches with extremely less FLOPs and lower latency.

8.3 Ablation Studies

This study conducts ablation experiments to evaluate various upsampling modules and loss function configurations for reducing computational demands and maintaining performance in visual tasks. It aims to identify efficient upsampling strategies and optimal loss combinations that preserve model accuracy. Experiments were standardized on the ADE20K validation set to ensure fair comparison and result reliability. The research compares the efficacy of techniques like bilinear interpolation, lightweight MobileNetV2-based upsampling, and standard convolutional upsampling, assessed by mIoU and computational impact. Adjusting loss functions, including classification loss, cross-model classification loss, feature similarity loss, and output similarity loss, further analyzes their role in knowledge distillation effectiveness. These experiments offer insights into balancing efficiency and performance in model design, providing a methodology for exploring model enhancements under computational constraints and guiding optimal configurations for accurate visual recognition in practical applications.

Table 17 Ablation study results of the upsampling modules

Upsampling Module	#Params	FLOPs	mIoU	Latency
Direct Interpolation	1.7M	0.3G	33.7	20ms
MobileNetV2-based upsample module	2.3M	0.3G	35.5	22ms
Standard convolution-based upsample module	2.7M	0.4G	35.7	30ms

Table 18 Ablation study results of loss function design

Teacher resolution	Cls loss	Out loss	Feat loss	Cross loss	mIoU
–	✓				32.5
512x512	✓	✓			33.7
512x512	✓	✓	✓		34.7
512x512	✓	✓	✓	✓	35.5
256x256	✓	✓	✓	✓	32.1

The resolution of the student model is 256 × 256

Impact of Upsample Module Design

We compare three different upsampling strategies: direct bilinear interpolation, MobileNetV2-based upsample module, and standard convolution-based upsample module.

In this section, the effects of three upsampling strategies on model performance were compared. The direct interpolation approach, while requiring the least parameters (1.7M) and computational effort (0.3G FLOPs), resulted in the lowest mIoU (33.7%) and the least latency (20ms), suggesting limited complexity handling. The MobileNetV2-based lightweight upsampling improved mIoU to 35.5% with a slight latency increase to 22ms, offering a balanced performance enhancement. The main components in the MobileNetV2-based upsampling module are 1×1 convolutional layers. From the table we conclude that 1×1 convolutional layers in the upsampling module contribute a minor increase in parameter count (+0.6M), which has a negligible impact on latency (+2ms) with a significant performance boost, from mIoU 33.7 to mIoU 35.5, demonstrating the effectiveness of our model design. The standard convolution-based module, although yielding the highest mIoU (35.7%), did so at the cost of increased parameters (2.7M), computation (0.4G FLOPs), and latency (30ms). These findings highlight a trade-off between performance and speed in upsampling choices, with the MobileNetV2-based module providing an optimal balance for dense prediction tasks on resource-constrained devices.

Impact of Loss Function

We incrementally add four loss function components-classification loss, output similarity loss, feature similarity loss, and cross-

model classification loss-to assess their contribution to model performance.

Through ablation studies, this experiment evaluates the performance impact of different loss functions in semantic segmentation, involving classification loss, output similarity loss, feature similarity loss, and cross-model classification loss, with mIoU as the evaluation metric. Starting from a baseline mIoU of 32.5 with just classification loss, performance sequentially improves with the addition of output and feature similarity losses, highlighting the benefits of aligning student and teacher model outputs and features for accuracy. The highest mIoU of 35.5 is achieved with the inclusion of cross-model classification loss, emphasizing the effectiveness of combining various constraints on the student model. The results underscore the individual and collective contributions of each loss function to semantic segmentation tasks, particularly the significance of model alignment for substantial performance gains, and the synergistic enhancement of model performance through integrated loss function design. To demonstrate the necessity and effectiveness of multi-resolution distillation. We compare with the conventional distillation baseline with a low-resolution teacher. As seen in the table, mIoU of the low-resolution teacher method is 32.1, which is significantly lower than the 35.5 achieved by our multi-resolution distillation method. This highlights the necessity and effectiveness of multi-resolution distillation for improving model performance.

9 Conclusion

In this paper, we propose squeeze-enhanced Axial Transformer (SeaFormer) for mobile semantic segmentation, filling the vacancy of mobile-friendly efficient Transformer. Moreover, we create a family of backbone architectures of SeaFormer and achieve cost-effectiveness. The superior performance on the ADE20K, Cityscapes Pascal Context and COCO-Stuff datasets, and the lowest latency demonstrate its effectiveness on the ARM-based mobile device. Moreover, we employ a feature upsampling-based multi-resolution distillation technique, significantly reducing the inference latency of our framework. This approach enhances model performance at various resolutions, enabling a high-resolution trained teacher model to instruct a low-resolution student model, thus facilitating efficient semantic understanding and prediction on mobile devices with reduced computational demands. Beyond semantic segmentation, we further apply the proposed SeaFormer architecture to image classification and object detection problems, demonstrating the potential of serving as a versatile mobile-friendly backbone.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (Grant No. 62376060).

Data Availability The datasets generated during and/or analysed during the current study are available in the Imagenet (Deng et al., 2009) (https://www.image-net.org/), COCO (Caesar et al., 2018) (https://cocodataset.org), ADE20K (Zhou et al., 2017) (https://groups.csail.mit.edu/vision/datasets/ADE20K/), Cityscapes (Cordts et al., 2016) (https://www.cityscapes-dataset.com), Pascal Context (Mottaghi et al., 2014) (https://cs.stanford.edu/~roozbeh/pascal-context/) and COCO-Stuff (Caesar et al., 2018) (https://github.com/nightrome/cocostuff?tab=readme-ov-file) repositories.

References

Caesar, H., Uijlings, J. & Ferrari, V. (2018). Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1209–1218.

Cao, Y., Xu, J., Lin, S., Wei, F. & Hu, H. (2019). Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0.

Chen, L.-C. (2017). Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587.

Chen, Y., Dai, X., Chen, D., Liu, M., Dong, X., Yuan, L. & Liu, Z. (2022). Mobile-former: Bridging mobilenet and transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5270–5279.

Chen, Y., Kalantidis, Y., Li, J., Yan, S. & Feng, J. (2018). A^2-nets: Double attention networks. *Advances in Neural Information processing systems, 31*.

Chen, Q., Wu, Q., Wang, J., Hu, Q., Hu, T., Ding, E., Cheng, J. & Wang, J. (2022). Mixformer: Mixing features across windows and dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5249–5259.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F. & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 801–818.

Cho, J.H. & Hariharan, B. (2019). On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4794–4802.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., *et al.* (2021). Rethinking attention with performers. In *International Conference on Learning Representations*.

Contributors, T. (2019). *TNN: A high-performance, lightweight neural network inference framework*.

Contributors, M. (2020). *MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark*.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. & Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3213–3223.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. & *et al.* (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Gao, H., Wang, Z. & Ji, S. (2020). Kronecker attention networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 229–237.

He, T., Shen, C., Tian, Z., Gong, D., Sun, C. & Yan, Y. (2019). Knowledge adaptation for efficient semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 578–587.

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

Heo, B., Kim, J., Yun, S., Park, H., Kwak, N. & Choi, J.Y. (2019). A comprehensive overhaul of feature distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1921–1930.

Hinton, G. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Ho, J., Kalchbrenner, N., Weissenborn, D. & Salimans, T. (2019). Axial attention in multidimensional transformers. arXiv preprint arXiv:1912.12180.

Hong, Y., Pan, H., Sun, W. & Jia, Y. (2021). Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes. arXiv preprint arXiv:2101.06085.

Hou, Q., Zhang, L., Cheng, M.-M. & Feng, J. (2020). Strip pooling: Rethinking spatial pooling for scene parsing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4003–4012.

Hou, Q., Zhou, D. & Feng, J. (2021). Coordinate attention for efficient mobile network design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13713–13722.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., *et al.* (2019). Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1314–1324.

Huang, Z., Ben, Y., Luo, G., Cheng, P., Yu, G. & Fu, B. (2021). Shuffle transformer: Rethinking spatial shuffle for vision transformer. arXiv preprint arXiv:2106.03650

Huang, Z., Wang, X., Huang, L., Huang, C., Wei, Y. & Liu, W. (2019). Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 603–612.

Huang, L., Yuan, Y., Guo, J., Zhang, C., Chen, X. & Wang, J. (2019). Interlaced sparse self-attention for semantic segmentation. arXiv preprint arXiv:1907.12273

Huang, Z., Wei, Y., Wang, X., Liu, W., Huang, T. S., & Shi, H. (2021). Alignseg: Feature-aligned segmentation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*(1), 550–557.

Hu, B., Zhou, S., Xiong, Z., & Wu, F. (2022). Cross-resolution distillation for efficient 3d medical image registration. *IEEE Transactions on Circuits and Systems for Video Technology, 32*(10), 7269–7283.

Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*.

Kim, J., Park, S. & Kwak, N. (2018). Paraphrasing complex network: Network compression via factor transfer. *Advances in Neural Information processing systems, 31*.

Kingma, D. P. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980

Kirillov, A., Girshick, R., He, K. & Dollár, P. (2019). Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6399–6408.

Li, J., Hassani, A., Walton, S. & Shi, H. (2023). Convmlp: Hierarchical convolutional mlps for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6307–6316.

Li, Y., Hu, J., Wen, Y., Evangelidis, G., Salahi, K., Wang, Y., Tulyakov, S. & Ren, J. (2023). Rethinking vision transformers for mobilenet size and speed. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16889–16900.

Li, Z., Li, X., Yang, L., Zhao, B., Song, R., Luo, L., Li, J. & Yang, J. (2023). Curriculum temperature for knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence, 37*, pp. 1504–1512.

Li, X., Li, X., Zhang, L., Cheng, G., Shi, J., Lin, Z., Tan, S. & Tong, Y. (2020). Improving semantic segmentation via decoupled body and edge supervision. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16*, pp. 435–452.

Li, H., Xiong, P., Fan, H. & Sun, J. (2019). Dfanet: Deep feature aggregation for real-time semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9522–9531.

Li, Z., Ye, J., Song, M., Huang, Y. & Pan, Z. (2021). Online knowledge distillation for efficient pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11740–11750.

Li, X., Li, X., You, A., Zhang, L., Cheng, G., Yang, K., Tong, Y., & Lin, Z. (2021). Towards efficient scene understanding via squeeze reasoning. *IEEE Transactions on Image Processing, 30*, 7050–7063.

Lin, T. (2017). Focal loss for dense object detection. arXiv preprint arXiv:1708.02002

Liu, Y., Chen, K., Liu, C., Qin, Z., Luo, Z. & Wang, J. (2019). Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2604–2613.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022.

Liu, P.J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L. & Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Liu, R., Yang, K., Roitberg, A., Zhang, J., Peng, K., Liu, H., Wang, Y. & Stiefelhagen, R. (2022). Transkd: Transformer knowledge distillation for efficient semantic segmentation. arXiv preprint arXiv:2202.13393

Li, Y., Yuan, G., Wen, Y., Hu, J., Evangelidis, G., Tulyakov, S., Wang, Y., & Ren, J. (2022). Efficientformer: Vision transformers at mobilenet speed. *Advances in Neural Information Processing Systems, 35*, 12934–12949.

Li, X., Zhang, L., Cheng, G., Yang, K., Tong, Y., Zhu, X., & Xiang, T. (2021). Global aggregation then local distribution for scene parsing. *IEEE Transactions on Image Processing, 30*, 6829–6842.

Long, J., Shelhamer, E. & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440.

Luong, M.-T. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025

Ma, N., Zhang, X., Zheng, H.-T. & Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 116–131.

Mehta, S. & Rastegari, M. (2022). Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. In *International Conference on Learning Representations*.

Mehta, S. & Rastegari, M. (2022). Separable self-attention for mobile vision transformers. arXiv preprint arXiv:2206.02680

Mirzadeh, S.I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A. & Ghasemzadeh, H. (2020). Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 5191–5198.

Mottaghi, R., Chen, X., Liu, X., Cho, N.-G., Lee, S.-W., Fidler, S., Urtasun, R. & Yuille, A. (2014). The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 891–898.

Pan, J., Bulat, A., Tan, F., Zhu, X., Dudziak, L., Li, H., Tzimiropoulos, G. & Martinez, B. (2022). Edgevits: Competing light-weight cnns on mobile devices with vision transformers. In *European Conference on Computer Vision*, pp. 294–311.

Pan, X., Ge, C., Lu, R., Song, S., Chen, G., Huang, Z. & Huang, G. (2022). On the integration of self-attention and convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 815–825.

Park, W., Kim, D., Lu, Y. & Cho, M. (2019). Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3967–3976.

Poudel, R.P., Liwicki, S. & Cipolla, R. (2019). Fast-scnn: Fast semantic segmentation network. arXiv preprint arXiv:1902.04502

Qi, L., Kuen, J., Gu, J., Lin, Z., Wang, Y., Chen, Y., Li, Y. & Jia, J. (2021). Multi-scale aligned distillation for low-resolution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14443–14453.

Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C. & Bengio, Y. (2014). Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520.

Shen, Z., Zhang, M., Zhao, H., Yi, S. & Li, H. (2021). Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3531–3539.

Tan, M. & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114.

Tang, S., Sun, T., Peng, J., Chen, G., Hao, Y., Lin, M., Xiao, Z., You, J. & Liu, Y. (2023). Pp-mobileseg: Explore the fast and accurate semantic segmentation model on mobile devices. arXiv preprint arXiv:2304.05152

Tian, Y., Krishnan, D. & Isola, P. (2019). Contrastive representation distillation. arXiv preprint arXiv:1910.10699

Vasu, P.K.A., Gabriel, J., Zhu, J., Tuzel, O. & Ranjan, A. (2023). Mobileone: An improved one millisecond mobile backbone. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7907–7917.

Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

Wan, Q., Huang, Z., Lu, J., Yu, G. & Zhang, L. (2023). Seaformer: Squeeze-enhanced axial transformer for mobile semantic segmentation. In *International Conference on Learning Representations*.

Wang, J., Chen, Y., Zheng, Z., Li, X., Cheng, M.-M. & Hou, Q. (2024). Crosskd: Cross-head knowledge distillation for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16520–16530.

Wang, S., Li, B.Z., Khabsa, M., Fang, H. & Ma, H. (2020). Linformer: Self-attention with linear complexity. arXiv preprint arXiv:2006.04768

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P. & Shao, L. (2021). Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 568–578.

Wang, H., Zhu, Y., Green, B., Adam, H., Yuille, A. & Chen, L.-C. (2020). Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European Conference on Computer Vision*, pp. 108–126.

Woo, S., Park, J., Lee, J.-Y. & Kweon, I.S. (2018). Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19.

Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M., & Luo, P. (2021). Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems, 34*, 12077–12090.

Xu, W., Xu, Y., Chang, T. & Tu, Z. (2021). Co-scale conv-attentional image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9981–9990.

Yan, H., Li, Z., Li, W., Wang, C., Wu, M. & Zhang, C. (2021). Contnet: Why not use convolution and transformer at the same time? arXiv preprint arXiv:2104.13497

Yang, C., Wang, Y., Zhang, J., Zhang, H., Wei, Z., Lin, Z. & Yuille, A. (2022). Lite vision transformer with enhanced self-attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11998–12008.

Yang, C., Xie, L., Su, C. & Yuille, A.L. (2019). Snapshot distillation: Teacher-student optimization in one generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2859–2868.

Yang, C., Zhou, H., An, Z., Jiang, X., Xu, Y. & Zhang, Q. (2022). Cross-image relational knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12319–12328.

Yim, J., Joo, D., Bae, J. & Kim, J. (2017). A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4133–4141.

Yu, C., Wang, J., Peng, C., Gao, C., Yu, G. & Sang, N. (2018). Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 325–341.

Yuan, Y., Chen, X. & Wang, J. (2020). Object-contextual representations for semantic segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pp. 173–190.

Yuan, Y., Fu, R., Huang, L., Lin, W., Zhang, C., Chen, X. & Wang, J. (2021). Hrformer: High-resolution transformer for dense prediction. arXiv preprint arXiv:2110.09408

Yu, C., Gao, C., Wang, J., Yu, G., Shen, C., & Sang, N. (2021). Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *International Journal of Computer Vision, 129*, 3051–3068.

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

Zhang, W., Huang, Z., Luo, G., Chen, T., Wang, X., Liu, W., Yu, G. & Shen, C. (2022). Topformer: Token pyramid transformer for mobile semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12083–12093.

Zhang, Y., Xiang, T., Hospedales, T.M. & Lu, H. (2018). Deep mutual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4320–4328.

Zhang, L., Xu, D., Arnab, A. & Torr, P.H. (2020). Dynamic graph message passing networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3726–3735.

Zhang, L., Chen, M., Arnab, A., Xue, X., & Torr, P. H. (2023). Dynamic graph message passing networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence, 45*(05), 5712–5730.

Zhao, B., Cui, Q., Song, R., Qiu, Y. & Liang, J. (2022). Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11953–11962.

Zhao, H., Qi, X., Shen, X., Shi, J. & Jia, J. (2018). Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 405–420.

Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P.H., *et al.* (2021). Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6881–6890.

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A. & Torralba, A. (2017). Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 633–641.