Simple and Efficient Vision Backbone Adapter for Image Semantic Segmentation

Dingjie Peng KEFIPHER9013@ASAGI.WASEDA.JP
Graduate School of Fund. Sci. and Eng., Waseda University, Tokyo, Japan
Wataru Kameyama WATARU@WASEDA.JP
Faculty of Sci. and Eng., Waseda University, Tokyo, Japan

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

Utilizing a pretrained vision backbone to finetune a model for semantic segmentation is common practice in computer vision. However, there are few works intending to enlarge the semantic context learning capacity by incorporating a segmentation adapter into the backbone. Thus, in this paper, we present a simple but efficient segmentation adapter, termed as SegAdapter, which can be plugged into the pretrained vision backbone to improve the performance of existing models for image semantic segmentation. We summarize SegAdapter with three attractive advantages: 1) SegAdapter is a plug-and-play module demonstrating strong adaptability in CNN and Transformer based models such as ConvNext and Segformer, 2) SegAdapter applies a light-weight High-order Spatial Attention (HSA) to make use of intermediate features from the pretrained backbone which extends the model depth and produces auxiliary segmentation maps for model enhancement, 3) SegAdapter builds a powerful vision backbone by incorporating the semantic context into each stage which takes on some of the functions of the segmentation head. So, SegAdapter augmented model can be used in simple designed decode head to avoid heavy computational cost. By plugging multiple SegAdapter layers into different vision backbones, we construct a series of SegAdapter-based segmentation models. We show through the extensive experiments that SegAdapter can be used with mainstream backbones like CNN and Transformer to improve mIoU performance in a large margin while introducing minimal additional parameters and FLOPs.

Keywords: Semantic Segmentation, Segmentation Adapter, CNN, Transformer, Spatial Attention.

1. Introduction

Semantic segmentation is a fundamental task in computer vision research which aims at assigning each pixel in an image to a specific category label. In the past few years, deep-learning-based semantic segmentation techniques have evolved considerably and shown impressive performance. Since much better results can be achieved by using pretrained models for semantic segmentation than by using ones that are trained from scratch, it becomes the de facto paradigm for modern deep learning training strategy. From Convolutional Neural Network (CNN) based models such as FCN (Long et al. (2015)), HRNet (Sun et al. (2019)) and ConvNext (Liu et al. (2022)), to Transformer-based models such as Pyramid Vision Transformer (Wang et al. (2021)), Swin Transformer (Liu et al. (2021)) and Segformer

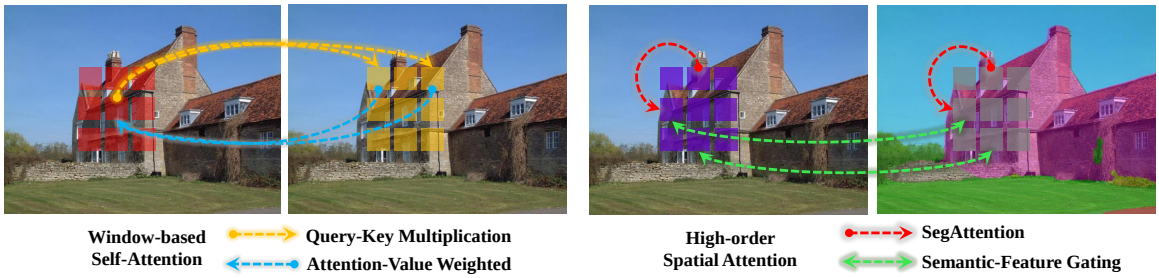


Figure 1: Simple illustrations of Window-based Self-Attention (WSA) and High-order Spatial Attention (HSA), showing the difference in colors. WSA applies query-key multiplication to obtain the attention score (■), which are used to perform attention-value weighted summation (■) without semantic context. HSA uses two SegAttention modules to encode regional information (■ and ■) with Depth-wise Separable Convolution (DSConv) of both input and segmentation map, respectively. The two features are further fused by semantic-feature gating to selectively incorporate the semantic context.

(Xie et al. (2021)), all of them tackle the semantic segmentation problem by employing the encoder-decoder architecture, where the encoder (backbone) is pretrained on a large-scale dataset (e.g., ImageNet (Deng et al. (2009))) for image classification task, then finetuning is applied to the entire model on the segmentation dataset.

However, simply using pretrained backbones to transfer downstream tasks in semantic segmentation usually leads to sub-optimal results. Jain et al. (2021) point out that the semantic context during the finetuning is very significant in segmentation task since this information can further enhance the model training for faster convergence. Hence, to allow better transfer of the pretrained backbone to semantic segmentation, they plug semantically masked attention into the hierarchical architectures to augment the global feature captured by the Transformers with the semantic context. However, this study is conducted only based on Swin Transformer backbone (Liu et al. (2021)), which cannot demonstrate its effectiveness on CNN-based backbones such as ConvNext (Liu et al. (2022)).

To address the above issues, we propose segmentation adapter (hereafter referred to SegAdapter) to encode the semantic context based on the Transformer and CNN backbone like Segformer (Xie et al. (2021))and ConvNext (Liu et al. (2022)). Specifically, we plug a SegAdapter layer at the end of each backbone layer, which uses High-order Spatial Attention (HSA) for sematic prior encoding. Figure 1 illustrates the difference between the Window-based Self-Attention (Liu et al. (2021)) and the proposed HSA. The former one applies query-key multiplication to obtain the attention map (weights), then the weights are used to perform attention-value weighted summation. So, no semantic context is involved in this module. However, HSA uses two SegAttention modules to encode regional information of both input and segmentation map, respectively, and the two features are further fused by semantic-feature gating. Hence, HSA can selectively incorporate the semantic context.

Our proposed SegAdapter can be plugged into any vision backbones for semantic segmentation. It can reduce the burden on the segmentation decode head while allowing the entire model to use a lighter decode head. Moreover, SegAdapter only introduces a small number of parameters and FLOPs while it can boost the mean Intersection over Union (mIoU) performance in a great margin compared with the baseline backbone. We summarize our contributions as follows:

- We propose SegAdapter as a plug-and-play module, which can be inserted into any existing pretrained backbone. It puts forward the semantic segmentation performance by encoding the semantic context and building the high-order spatial attention.
- Through extensive experiments, we show that SegAdapter achieves a considerable improvement in semantic segmentation for ADE20K (Zhou et al. (2017)), Cityscapes (Cordts et al. (2016)) and COCO-Stuff (Caesar et al. (2018)). Especially on ADE20K task, SegAdapter can improve the Segformer (Xie et al. (2021)) and ConvNext (Liu et al. (2022)) up to 2.9% and 3.5% in mIoU, respectively.
- Our ablation studies elaborate more evidence for the effective and efficient design of the SegAdapter, which provides additional experimental data and inspiration for modern adapter designs.

2. Related Works

2.1. Semantic Segmentation

FCN (Long et al. (2015)) is the first model to use convolution to encode images into abstract features and to decode them into the segmentation maps without applying the fully connected layers. Following that, many convolution-based approaches have been developed. DeepLab v3+ (Chen et al. (2018)) improves the Atrous Spatial Pyramid Pooling (ASPP), and incorporates it into the decoder to further enlarge the receptive field. Recently, many researchers have proposed Transformer architecture for computer vision. Vision Transformer (ViT) (Dosovitskiy et al. (2020)) is a pioneer that applies the vanilla Transformer on the image classification task. However, since non-hierarchical architecture is unsuitable for dense prediction tasks, Liu et al. (2021) proposes Swin Transformer that uses Shifted Window based Self-Attention (W-MSA) and patch merging to reduce the computational cost to form a hierarchical framework. In this paper, we follow these previous works using pretrained backbones, but we focus on building a powerful and efficient segmentation adapter.

2.2. Adapters for Semantic Segmentation

Applying adapter to a pretrained model allows backbone to incorporate task-relevant prior knowledge to speed up the convergence. Jain et al. (2021) argue that the absence of semantic context during encoding may lead to a poor segmentation performance. So, they propose to insert a semantic layer after each Swin Transformer stage to use semantically masked feature. Vision Transformer Adapter (ViT-Adapter) (Chen et al. (2022)) is designed specifically to help the ViT adapting to the downstream dense prediction task. It uses the Deformable Self-Attention (Xia et al. (2022)) as the attention mechanism, then fuses the multi-scale adapter

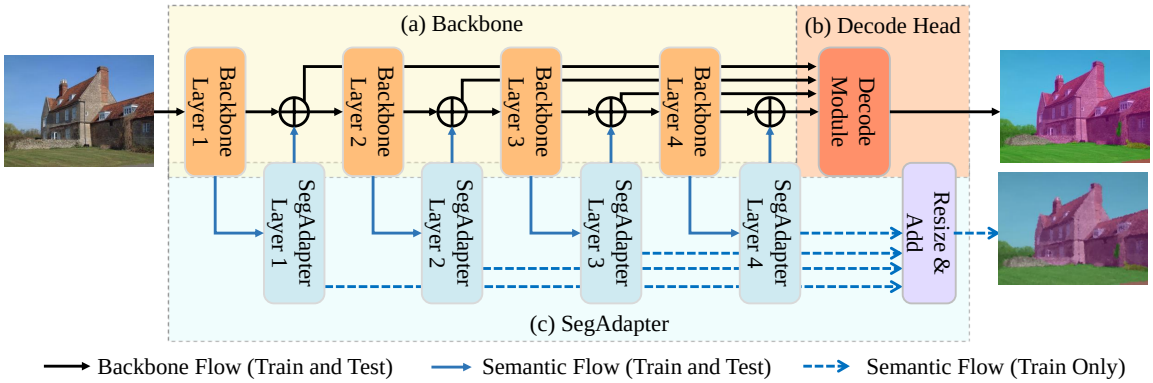


Figure 2: Overall architecture of SegAdapter for semantic segmentation.

features and backbone features by cross attention. Instead of designing a segmentation adapter for a certain backbone, we propose SegAdapter that can be plugged into any existing models, which shows the effectiveness through extensive experiments.

2.3. Attention Mechanisms

The attention mechanism works by assigning a weight to each input feature, which determines how much attention the model should pay to that feature when making predictions. Vaswani et al. (2017) propose the first work to introduce the self-attention for language translation. After that, various self-attention mechanisms have been used in a variety of computer vision fields. For example, DETR (Xia et al. (2022)) proposes Deformable Self-Attention for object detection, Segformer (Xie et al. (2021)) presents Efficient Self-Attention to down-sample the key and value for semantic segmentation, and CrossViT (Chen et al. (2021)) utilizes cross-attention to learn multi-scale features for vision. As describe in Rao et al. (2022), the above works are the variants of self-attention as the type of 3-order spatial interaction. However, we build a 4-order spatial interaction via two simple and efficient SegAttention modules.

3. Proposal

3.1. Overall Architecture

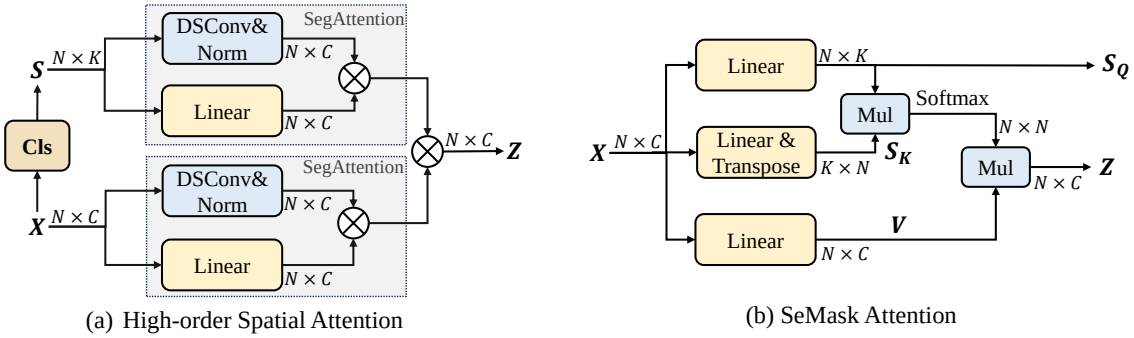
As shown in Figure 2, there are three main components in the proposed SegAdapter augmented model: (i) a backbone which can be the CNN-based architecture (e.g., ConvNext (Liu et al. (2022))) or the Transformer-based architecture (e.g., Segformer (Xie et al. (2021))) to extract the abstract features from the input image, (ii) a decode head that fuses the features from different backbone stages and produces the final prediction, (iii) a SegAdapter that absorbs the multi-scale output from backbone and augments them by semantic context at the end of each stage or layer.

In detail, the output of backbone layer-i is fed to the SegAdapter layer-i ($i = 1, 2, 3, 4$) to produce a side output prediction via a simple classification head, where the output of

backbone layer is augmented by the proposed High-order Spatial Attention (HSA). After that, we inject the augmented output back to the vision backbone with a small scale μ. The above-mentioned process is denoted as "semantic flow" with blue arrow in Figure 2. During training, the side output predictions from the four stages are all resized into the 1/4 input size, and then they are added together to produce a coarse segmentation map, which is supervised by the per-pixel cross-entropy loss. The auxiliary path for producing the coarse segmentation map works as another "semantic flow" that can be removed during testing, which is denoted as blue dashed arrow in Figure 2.

3.2. Segmentation Adapter

In this section, we first briefly recap the mechanisms of Multi-head Self-Attention (Dosovitskiy et al. (2020)) and its variant, SeMask Attention (Jain et al. (2021)), which is specifically designed for the Transformer backbone. Next, we cover the proposed SegAttention and the main component of SegAdapter: High-order Spatial Attention.



(a) High-order Spatial Attention (b) SeMask Attention

Figure 3: The architecture of High-order spatial attention and SeMask attention.

Self-Attention and SeMask Attention. Self-attention involves computing three matrices: query, key and value. Basically, query and key are used to generate a set of spatial attention weights via matrix multiplication ("Mul" in Figure 3-(b)), then the weights are used to compute a weighted sum of the value matrix. Most methods divide the embedding channels C into $\lceil C/d \rceil$ groups as Multi-head Self-Attention (MSA) (Dosovitskiy et al. (2020)), where each head can be formulated as follows:

$$\boldsymbol{Z} = \text{Softmax}\left(\boldsymbol{Q}\boldsymbol{K}^T/\sqrt{d}\right)\boldsymbol{V}, \tag{1}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are query, key and value learned from input $X \in \mathbb{R}^{N\times d}$, N denotes the token length (same as input size in vision task), and d presents the channel dimension of each head. Liu et al. (2021) introduce Shifted Window based Multi-head Self-Attention (W-MSA) to reduce the computational complexity of MSA by restricting the self-attention inside a local window. After the seminal work of Liu et al. (2021), SeMask Attention (Jain et al. (2021)) is proposed to encode the semantic context based on W-MSA. As shown in Figure 3-(b), SeMask generates a segmentation map $\boldsymbol{S_Q}$ as query to perform W-MSA as follows:

$$\boldsymbol{Z} = \text{Softmax}\left(\boldsymbol{S_Q}\boldsymbol{S_K}^T\right)\boldsymbol{V}, \tag{2}$$

where $\boldsymbol{S_Q}, \boldsymbol{S_K} \in \mathbb{R}^{N\times K}$, $\boldsymbol{V} \in \mathbb{R}^{N\times C}$ are learned by linear transformation based on input, K denotes the number of classes, and C denotes the embedding dimension. Note that in Eq. (2), the side output $\boldsymbol{S_Q}$ is the segmentation map exactly guided by the cross-entropy loss during training.

SegAttention. We argue that it is sub-optimal for Semask (Jain et al. (2021)) to simply utilize side output prediction as query in adapter, because SeMask Attention produces the semantic query $\boldsymbol{S_Q}$ and semantic key $\boldsymbol{S_K}$ simultaneously, but only $\boldsymbol{S_Q}$ is supervised by the per-pixel loss, which may cause unclear meaning for spatial semantic attention for $\boldsymbol{S_K}$. Besides, VAN (Guo et al. (2022)) points out that the simple convolutional attention can be more efficient to maintain the local continuity when compared to the complex design of W-MSA. Therefore, for the attention design, we construct a simple but effective convolutional-based spatial attention and termed it as SegAttention. The illustration of SegAttention is shown in Figure 3-(a) with a gray box. We produce attention score $\boldsymbol{U}$ by applying Depth-wise Separable Convolution (DSConv) with the kernel size of $s \times s$ and Layer Normalization (Norm) (Ba et al. (2016)) based on input $\boldsymbol{X}$. Here, $\boldsymbol{U}$ considers an area of $s \times s$ as receptive field for one pixel in $\boldsymbol{X}$. Meanwhile, the value $\boldsymbol{V}$ is obtained by linear transformation. We employ Hadamard product ($\otimes$) to perform spatial attention based on $\boldsymbol{U}$ and $\boldsymbol{V}$. So, the proposed SegAttention module can be formulated as follows:

$$\begin{aligned} \boldsymbol{U} &= \text{Norm}\left(\text{DSConv}(\boldsymbol{X})\right), \\ \boldsymbol{V} &= \text{Linear}(\boldsymbol{X}), \\ \boldsymbol{F} &= \text{SegAttention}(\boldsymbol{X}) = \boldsymbol{U} \otimes \boldsymbol{V}. \end{aligned} \tag{3}$$

High-order Spatial Attention. As depicted in Figure 3-(a), we present a High-order Spatial Attention (HSA) for the interaction between the backbone output and the side output prediction. Similar to Jain et al. (2021), the backbone output $\boldsymbol{X} \in \mathbb{R}^{N\times C}$ is first reshaped into the size of $C \times H \times W$, where $N = H \times W$. And then $\boldsymbol{X}$ is fed into the 1×1 convolutional classification head (denoted as "Cls" in Figure 3-(a)) to generate a side output prediction $\text{Cls}(\boldsymbol{X}) \in \mathbb{R}^{K\times H\times W}$. Then, the SegAttention is applied as the 2-order spatial interaction to the backbone output $\boldsymbol{X}$ and the side output prediction $\text{Cls}(\boldsymbol{X})$ separately. After that, the outputs of the two SegAttention, $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$, are further fused by Hadamard product ($\otimes$) to generate augmented feature $\boldsymbol{Z}$, which simply encodes the semantic context into the attention mechanism. Thus, the HSA process can be formulated as follows:

$$\begin{aligned} \boldsymbol{F}_1 &= \text{SegAttention}(\boldsymbol{X}), \\ \boldsymbol{F}_2 &= \text{SegAttention}\left(\text{Cls}(\boldsymbol{X})\right), \\ \boldsymbol{Z} &= \text{HSA}(\boldsymbol{X}) = \boldsymbol{F}_1 \otimes \boldsymbol{F}_2. \end{aligned} \tag{4}$$

Note that this process can be also considered as pixel attention (Guo et al. (2022)) or feature gating (Rao et al. (2022)), since we use pixel-wise multiplication to fuse the feature attention and semantic context attention. From another view, the HSA can be also expressed as follows:

$$\boldsymbol{Z} = \text{HSA}(\boldsymbol{X}) = \Phi_1(\boldsymbol{X}) \otimes \Psi_1(\boldsymbol{X}) \otimes \Phi_2(\text{Cls}(\boldsymbol{X})) \otimes \Psi_2(\text{Cls}(\boldsymbol{X})), \tag{5}$$

where $\Phi_1(\cdot)$ and $\Phi_2(\cdot)$ denote the linear transform functions for the feature and semantic space, respectively. $\Psi_1(\cdot)$ and $\Psi_2(\cdot)$ present the combination of DSConv$(\cdot)$ and Norm$(\cdot)$. Thus, from Eq. (5), the proposed HSA is a 4-order operation spatial attention.

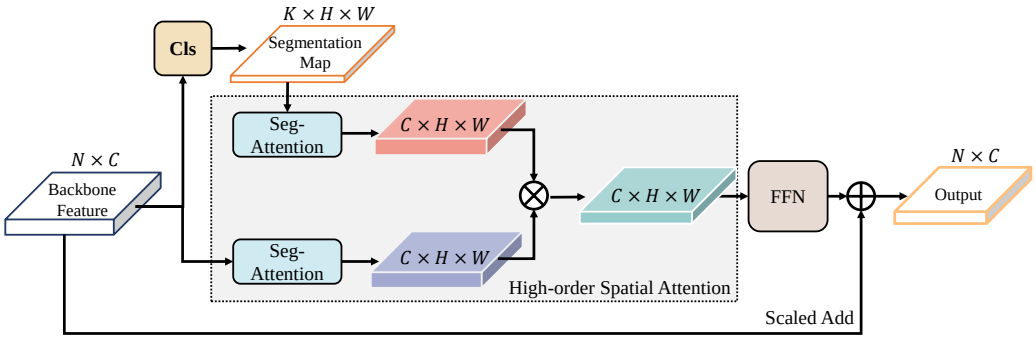


Figure 4: The architecture of SegAdapter block.

SegAdapter Layer. Similar to other backbone layers such as Sun et al. (2019) and Woo et al. (2023), we can stack multiple SegAdapter blocks to form a SegAdapter layer. As illustrated in Figure 4, the SegAdapter block includes an HSA and a standard Feed Forward Network (FFN). For the residual learning, we use a learnable parameter $\mu \in \mathbb{R}^C$ for each channel to avoid changing the backbone output dramatically as follows:

$$\boldsymbol{Y} = \boldsymbol{X} + \mu \cdot \mathrm{FFN}(\boldsymbol{Z}), \tag{6}$$

where $\boldsymbol{Y} \in \mathbb{R}^{N \times C}$ denotes the output of the SegAdapter block, $\boldsymbol{X} \in \mathbb{R}^{N \times C}$ denotes the backbone output, and $\boldsymbol{Z} \in \mathbb{R}^{N \times C}$ is the output of HSA which is fed to the FFN to fuse channel information.

3.3. Computational Complexity Analysis

In this section, we first recap the computational complexity of MSA (Dosovitskiy et al. (2020)), W-MSA (Liu et al. (2021)) and SeMask (Jain et al. (2021)), then introduce our SegAttention for investigation.

From Eq. (1), with the definitions of N, H, W, C and K in Section 3.2, the computational complexity of MSA is formulated as follows:

$$\Omega(\mathrm{MSA}) = 4HWC^2 + 2(HW)^2C. \tag{7}$$

From above, the computational cost of MSA is quadratic to the input size $N = H \times W$, which is unsuitable for dense prediction task. Hence, Liu et al. (2021) introduces W-MSA, which first evenly partitions the input into multiple windows of size $M \times M$ in a non-overlapping manner, then it computes self-attention within local windows. Thus, the complexity of W-MSA is:

$$\Omega(\text{W-MSA}) = 4HWC^2 + 2HWCM^2, \tag{8}$$

which is linear to the input size N. Considering that W-MSA does not integrate semantic information into backbone, SeMask (Jain et al. (2021)) presents a variant of W-MSA by producing the query and value in the semantic space, and keeps other parts unchanged. Hence, the complexity of SeMask Attention can be obtained as follows:

$$\begin{aligned} \Omega(\mathrm{SeMask}) &= 2HWCK + 2HWC^2 + HWKM^2 + HWCM^2 \\ &= HW(2C + M^2)(C + K). \end{aligned} \tag{9}$$

From Eq. (2), both W-MSA and SeMask Attention are 3-order spatial attention since they follow the standard self-attention manner that depicted in Eq. (1).

Inspired by Jain et al. (2021), the proposed HSA uses backbone output to generate a side output prediction, and it applies two SegAttention branches to construct spatial attention for the feature and the semantic space separately. So, from Eq. (3), the complexity of HSA is formulated as follows:

$$\begin{aligned}\Omega(\text{HSA}) &= 2HWCK + 3HWC^2 + s^2HWC + s^2HWK + 3HWC \\ &= HW(2CK + 3C^2 + s^2C + s^2K + 3C).\end{aligned} \tag{10}$$

In the experiments, we set the kernel size of SegAttention to 5×5 by default. When using the window size of 7×7 as default in SeMask (Jain et al. (2021)), we can obtain the computational cost ratio r between Ω(SeMask) and Ω(HSA) as follows:

$$r = \frac{\Omega(\text{SeMask})}{\Omega(\text{HSA})} = \frac{(2C + 49)(C + K)}{2CK + 3C^2 + 25(C + K) + 3C}. \tag{11}$$

In the first stage, the embedding dimension C is 64, the number of classes K is 150 for ADE20K dataset (Zhou et al. (2017)), and r is around 1.02. The ratio r decreases to 0.74 when C is set to 512 in the final stage. Therefore, our HSA can achieve a 4-order spatial interaction with the similar computational cost as in SeMask, which is a 3-order one for adapter.

3.4. Decode Head

We induce the light-weight decode head in Xie et al. (2021), which uses simple MLP to align the number of channels to C_d for multi-scale features from the four stages of backbone. After MLP, these features are concatenated along the channel dimension, and then projected to produce the main prediction $\boldsymbol{S_m}$.

3.5. Training Objective

Our proposed SegAdapter produces two segmentation maps: main prediction $\boldsymbol{S_m}$ from decode head, and the coarse output $\boldsymbol{S_c}$ from Adapter. We use cross-entropy loss $\mathcal{L}_{ce}$ to guide these two outputs for supervised training. Following the strategy in Jain et al. (2021), the overall training objective $\mathcal{L}$ is the weighted sum of the two losses as follows:

$$\mathcal{L} = \mathcal{L}_{ce}(\boldsymbol{S_m}, \boldsymbol{G}) + \lambda \cdot \mathcal{L}_{ce}(\boldsymbol{S_c}, \boldsymbol{G}), \tag{12}$$

where $\boldsymbol{G}$ denotes the ground truth segmentation map, λ is a hyper parameter for trading-off the main prediction loss $\mathcal{L}_{ce}(\boldsymbol{S_m}, \boldsymbol{G})$ and the coarse prediction loss $\mathcal{L}_{ce}(\boldsymbol{S_c}, \boldsymbol{G})$.

3.6. Model Configurations

In practice, we set the FFN ratio r_f of SegAdapter to 3.0, the number of SegAdapter blocks N_b to $[1, 1, 1, 1]$ and the learnable parameter μ to $[10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$ for four stages unless it specifies. The channel dimension C_d in decode head is fixed to 256 for all models unless other specified. And the trade-off loss parameter λ is set to 0.4 as default.

4. Experiments

4.1. Datasets

Following the benchmark of the mainstream semantic segmentation task, we use three datasets: ADE20K (Zhou et al. (2017)), Cityscapes(Cordts et al. (2016)) and COCO-Stuff (Caesar et al. (2018)). The ADE20K dataset covers 150 semantic categories, including 20,210 images for training and 2,000 images for validation. Cityscapes provides pixel annotations on 5,000 high resolution images with 19 categories in total. COCO-Stuff dataset is constructed with 164k images that span over 172 categories, where 118k for training, 5k for validation, 20k for the test-challenge and 20k for test-dev.

4.2. Implementation Details

In experiments, we select two state-of-the-art vision backbones for our SegAdapter: MIT (the backbone of Segformer (Xie et al. (2021))) and ConvNext (Liu et al. (2022)). The backbones pretrained on ImageNet 1K dataset (Deng et al. (2009)) are from the public source of Openmmlab (Contributors (2020)), where we keep the backbone configurations unchanged, then build the SegAdapter models based on them. Basically, we refer SegAdapter models to "Adapt-*backbone-variant*", for example, Segformer-B0 with SegAdapter augmentation is named as "Adapt-MIT-B0". Following other segmentation methods as Liu et al. (2022), Cheng et al. (2021) and Xia et al. (2022), we resize images with the ratio of [0.5, 2.0], and apply random horizontal flip and random cropping. We use the AdamW optimizer (Loshchilov and Hutter (2017))with initial learning rate of 6×10^{-5}, and the poly schedule with the power of 1. The batch size is set to 8 for Cityscapes (Cordts et al. (2016)), and 16 for other datasets. Note that Segformer (Xie et al. (2021)) uses crop size of 1024×1024 for training on Cityscapes while we modify it to 768×768 for saving the GPU memory. We train all models for 160K iterations on ADE20K (Zhou et al. (2017)) and Cityscapes (Cordts et al. (2016)), 80K iterations on COCO-Stuff (Caesar et al. (2018)), and use the same testing pipeline as same in Xie et al. (2021). For all three datasets, we report the single scale testing of mean Intersection over Union (mIoU).

We conduct extensive experiments to evaluate the effectiveness of the proposed SegAdapter on mainstream backbones like CNN and Transformer. For fair comparisons, we use the MLP decode head in Xie et al. (2021), and set the decode channels $C_d = 256$ for all models. However, note that the original Segformer-B2 and Segformer-B3 (Xie et al. (2021)) use a large size decode head with $C_d = 768$ while we use $C_d = 256$ for SegAdapters. So, to distinguish the different size of decode heads, the Segformer method (B2 or B3) with $C_d = 256$ is denoted as Segformer* and that with $C_d = 768$ is denoted as Segformer** in the following experiments.

4.3. Results

Through Table 1, on ADE20K (Zhou et al. (2017)), SegAdapter raises mIoU by 2.9% compared to the tiny baseline models (Segformer-B0 and ConvNext-T), and over 1.4% compared to small models (Segformer-B1, B2 and ConvNext-S). For base models, Segformer-B3 and ConvNext-B, SegAdapter suppress them by 0.8% and 1.9% in mIoU, respectively. On Cityscapes (Cordts et al. (2016)), SegAdapter boosts mIoU by 0.5-2.7%. On COCO-stuff

Table 1: mIoU (%) and FLOPs (G) comparisons of baselines and corresponding SegAdapters on ADE20K, Cityscapes, and COCO-Stuff semantic segmentation. FLOPs are counted as input size 512^2 for ADE20K and COCO-Stuff, and 1024^2 for Cityscapes, respectively.

Model	ADE20K		Cityscapes		COCO-Stuff	
	mIoU(%)	FLOPs(G)	mIoU(%)	FLOPs(G)	mIoU(%)	FLOPs(G)
Segformer-B0	37.4	8.4	76.2	44.0	35.6	8.4
Adapt-MIT-B0	**40.3** (+2.9)	9.7	**77.8** (+1.6)	47.0	**36.6** (+1.0)	9.7
Segformer-B1	42.2	15.9	78.1	86.6	40.2	15.9
Adapt-MIT-B1	**44.0** (+1.8)	19.5	**79.6** (+1.5)	97.4	**41.1** (+0.9)	19.5
Segformer-B2*	45.9	25.8	81.0	147.7	44.6	25.8
Adapt-MIT-B2	**48.0** (+2.1)	29.4	**81.5** (+0.5)	158.6	**45.5** (+0.9)	29.4
Segformer-B3*	48.6	42.5	81.4	238.6	45.0	42.5
Adapt-MIT-B3	**49.4** (+0.8)	46.1	**81.9** (+0.5)	249.5	**46.2** (+1.2)	46.1
ConvNext-T	41.6	29.0	77.2	113.7	39.7	29.0
Adapt-ConvNext-T	**45.1** (+3.5)	32.9	**79.9** (+2.7)	134.6	**42.0** (+2.3)	32.9
ConvNext-S	45.6	51.1	79.9	202.1	41.6	51.1
Adapt-ConvNext-S	**47.0** (+1.4)	57.5	**80.6** (+0.7)	223.0	**43.1** (+1.5)	57.5
ConvNext-B	46.1	86.2	80.8	342.5	42.4	86.2
Adapt-ConvNext-B	**48.0** (+1.9)	96.9	**81.6** (+0.8)	379.0	**44.6** (+2.2)	96.9

(Caesar et al. (2018)), we also observe remarkable performance of SegAdapter with 0.9-2.3% improvements in mIoU. These results demonstrate that SegAdapter is a viable solution for improving the performance of existing CNN or Transformer backbones for semantic segmentation.

Table 2: Comparison with state-of-the-art methods on ADE20K dataset.

Method	Backbone	Params (M)	FLOPs (G)	mIoU (%)
SeMask	MIT-B0	4.7	9.8	39.1
SegAdapter	MIT-B0	4.8	**9.7**	**40.3**
SeMask	MIT-B1	17.1	20.7	42.7
SegAdapter	MIT-B1	17.3	**19.5**	**44.0**
Segformer**	MIT-B2	27.5	62.4	46.5
MaskFormer	Swin-T	42.0	55.0	46.7
SeMask	MIT-B2	28.2	30.6	47.0
SegAdapter	MIT-B2	28.4	**29.4**	**48.0**
SeMask	MIT-B3	37.2	47.2	48.3
HRFormer-B	HRFormer	56.2	280.0	48.7
Swin-B FPN	Swin-B	93.0	103.0	48.8
Segformer**	MIT-B3	47.3	79.0	**49.4**
SegAdapter	MIT-B3	37.4	**46.1**	**49.4**

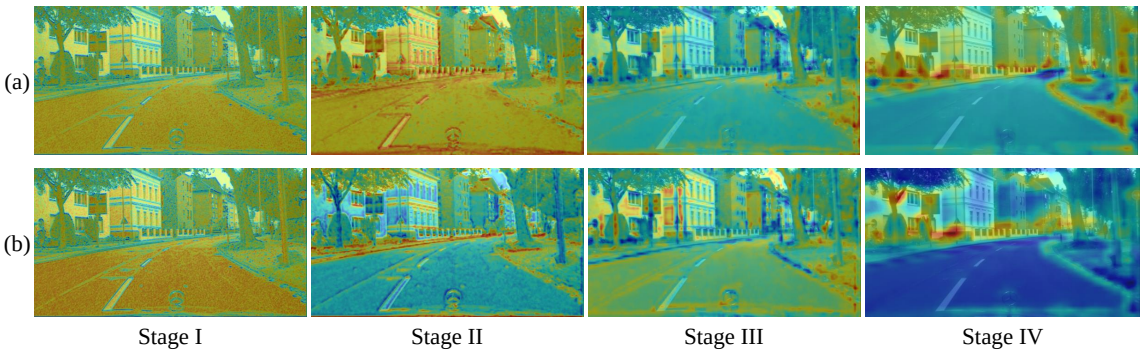


Figure 5: Stage feature visualization of (a) Segformer-B0 and (b) Adapt-MIT-B0.

We also compare SegAdapter with SeMask (Jain et al. (2021)), another adapter strategy for semantic segmentation. In the experiments, we replace HSA with SeMask Attention while keeping other modules unchanged and training on ADE20K (Zhou et al. (2017)). For a more comprehensive comparison, in Table 2, we also list other state-of-the-art methods, such as HRformer (Yuan et al. (2021)), MaskFormer (Cheng et al. (2021)) and Segformer** (Xie et al. (2021)) in the original paper. From the table, when using similar model size ("Params" in Table 2) and slightly smaller computational cost (FLOPs), our SegAdapter can surpass SeMask by more than 1% in mIoU, which indicates the strong learning capacity of the SegAdapter. Moreover, SegAdapter surpasses other non-adapter methods such as HRFormer and MaskFormer in a great margin even though we have much smaller model parameters and FLOPs. Also, we notice that in Segformer, the decode head with $C_d = 768$ causes a large number of FLOPs (See Segformer* in Table 1 and Segformer** in Table 2). However, the proposed Adapt-MIT-B3 can still perform on-par with Segformer-B3** even though we have a much smaller FLOPs when using $C_d = 256$.

4.4. Visualization

To better illustrate the differences between our proposed SegAdapter and Segformer, we visualize features of the four stages of the Adapt-MIT-B0 and Segformer-B0 (Xie et al. (2021)) for comparison as shown in Figure 5. The input is from Cityscapes validation set (Cordts et al. (2016)), and we reshape feature outputs from the four stages to the original input size with visualizing them as heatmaps. From the figure, the Stage I features are very similar because we use a small μ to control the semantic context learning. After that, the two models exhibit very different learning patterns: in Stage II, Segformer-B0 shows a very high response globally, especially on "roads" and "buildings", while Adapt-MIT-B0 exhibits a high response only on object contours. In Stage III, Adapt-MIT-B0 focuses more on "trunks" and "walls" compared to Segformer-B0, which indicates that Adapt-MIT-B0 incorporates more semantic context in deeper layers. In Stage IV, Adapt-MIT-B0's low response area is primarily concentrated on "roads" and "leaves," and it shows greater contrast than Segformer-B0. This indicates that Adapt-MIT-B0 exhibits more prominent semantic feature contrasts in the final layers.

The ground truth and segmentation outputs are shown in Figure 6. We zoom in on the semantic segmentation result of a pedestrian in the distance for comparison. As we can observe, Segformer-B0 (Xie et al. (2021)) fails to accurately segment the far-away pedestrian, while Adapt-MIT-B0, which incorporates semantic context into attention of backbone, performs better in segmenting the outlines of the pedestrian.

4.5. Ablation Studies

In this section, we conduct several experiments on ADE20K (Zhou et al. (2017)) to ablate the components of our SegAdapter. In detail, the ablation experiments include the kernel size $s \times s$, normalization layer of SegAttention, FFN ratio r_f in SegAdapter block, the number of SegAdapter blocks N_b for each stage, as well as the initial value of the trade-off parameter μ. We use MIT-B1 as backbone and MLP decode head with channel dimension $C_d = 256$ for all experiments.

Table 3: Ablation on (a) Kernel size of SegAttetion, (b) FFN ratio of SegAdapter block and (c) Normalization type of SegAttention.

(a) Kernel size.

kernel size	mIoU(%)
3×3	43.8
5×5	44.0
7×7	44.1

(b) FFN ratio.

FFN ratio	mIoU(%)
2.0	43.6
3.0	44.0
4.0	44.1

(c) Normalization type.

Normalization	mIoU(%)
Layer Norm	44.0
Batch Norm	43.5
Instance Norm	43.1

Kernel size of SegAttention. We evaluate the effectiveness of kernel size s in SegAttention by setting $s = 3$, 5 and 7. From Table 3-(a), we can observe that increasing the convolution kernel sizes can expands the receptive field to capture more context in a local region, thus resulting in a slight increase in mIoU. We notice that larger kernel size causes slower train and inference speed, so we use 5×5 kernel size as default for trading-off.

FFN ratio in SegAdapter Block. For an attention block, it is natural to place a two-layer FFN after the attention module, which is pointed out as vital for modern block design (Yu et al. (2022)). In ablation, we investigate FFN ratio r_f in the dense fully connected layers by setting r_f to 2.0, 3.0 and 4.0 in the experiments. From Table 3-(b), as r_f becomes

Ground Truth Segformer-B0 Adapt-MIT-B0

Figure 6: Visual result comparisons of Ground Truth, Segformer-B0 and Adapt-MIT-B0.

larger, we observe a stable improvement in mIoU performance. However, the situation saturates when $r_f = 4.0$. Thus, considering the efficiency, we use $r_f = 3.0$ by default.

Normalization of SegAttention. We use Layer Normalization (LN) (Ba et al. (2016)) by default since almost all self-attention mechanisms in the community apply this normalization type. Therefore, in ablation, we also employ Batch Normalization (BN) (Ioffe and Szegedy (2015)) and Instance Normalization (IN) (Ulyanov et al. (2016)) to evaluate the impact of different normalization layers in SegAttention. From Table 3-(c), if we apply BN, the mIoU decreases to 43.5%. However, when using LN, we obtain the best performance of 44.0% in mIoU, which is consistent with that in the ConvNext (Liu et al. (2022)): after replacing BN with LN, they observe slight performance improvements. This is because both Segformer and ConvNext use LN in the backbone, that implicitly encourages the feature to act differently along channels. Finally, the IN based model reaches only 43.1% in mIoU, which indicates that only normalizing the token length (height and width of the feature) on a single sample is inappropriate for the Adapter.

Table 4: Ablation on the (a) initial value of μ and (b) number of SegAdapter blocks.

(a) Initial value of μ.

Initial value of μ	mIoU(%)
const $[1.0, 1.0, 1.0, 1.0]$	43.2
$[1.0, \ 1.0, \ 1.0, \ 1.0]$	43.4
$[10^{-2}, 10^{-2}, 10^{-2}, 10^{-2}]$	43.7
$[10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}]$	43.8
$[10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}]$	43.4
$[10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$	44.0

(b) Number of SegAdapter blocks.

# Blocks	FLOPs (G)	mIoU (%)
$[0, 0, 0, 0]$	15.9	42.2
$[1, 1, 1, 1]$	19.5	44.0
$[1, 1, 2, 1]$	20.6	43.8
$[1, 2, 2, 1]$	21.4	43.9
$[1, 2, 4, 1]$	23.4	43.8
$[1, 2, 6, 1]$	25.4	44.1

Initial value of the trade-off parameter μ. We ablate the parameter μ in Eq. (6), which is used to control the amount of semantic context information injection of the backbone. As shown in Table 4-(a), By keeping the $\mu = [1, 0, 1.0, 1.0, 1.0]$ as constant (not learnable) for all four stages, we obtain a mIoU of 43.2%, which is 1.0% higher than the baseline Segformer-B1 (42.0%). However, when $\mu = [1.0, 1.0, 1.0, 1.0]$ are learnable, the mIoU result increases to 43.4%, which demonstrates that setting μ learnable can boost the performance. Moreover, we find out that initializing μ with a small value, e.g., $\mu = 10^{-2}$ and $\mu = 10^{-4}$ for all stages, can further increase the performance to 43.7% and 43.8%, respectively. Intuitively, as the layer goes deeper, it is encouraged to gradually increase initial μ for better incorporating more semantic context. Therefore, when $\mu = [10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$ are learnable, we observe the highest mIoU as 44.0% in our study. If we reverse the initial value as $\mu = [10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}]$, The performance drop to 43.4%, which indicates that the effectiveness increases by setting the trade-off parameter μ learnable as the network deepens.

Number of SegAdapter Blocks. We study the influence of the Number of SegAdapter blocks N_b for each stage. Here, $N_b = [0, 0, 0, 0]$ represents the Segformer-B1 baseline model. We notice that the modern design of CNN-based and Transformer-based vision backbones (Xie et al. (2021), Yu et al. (2022)) usually assign a few blocks for Stage II, and assign a large number of blocks for Stage III to build a larger model. Inspired by these, we ablate the

number of SegAdapter blocks N_b by adding the SegAdapter blocks to the second and third stage as $N_b = [1, 1, 2, 1]$, $[1, 2, 2, 1]$, $[1, 2, 4, 1]$ and $[1, 2, 6, 1]$. Through Table 4-(b), we get same conclusion as in SeMask (Jain et al. (2021)): i.e., the best setting is $N_b = [1, 1, 1, 1]$, and further increasing the SegAdapter block does not bring performance gain. It indicates that stacking too many random initialized blocks to the pretrained backbone may disrupt the distribution of the original feature.

5. Conclusion

In this paper, we propose SegAdapter, an efficient segmentation adapter to improve the performance of the existing backbones for image semantic segmentation without requiring additional pretraining. Our experiments show that SegAdapter can perform as a plug-and-play module with the mainstream backbones like CNN and Transformer. The novel High-order Spatial Attention constructs an adaptive selection on both features and segmentation map, and fuses them via Hadamard product, which dynamically encodes the semantic context into the backbone. Besides, the process of generating segmentation map in SegAdapter layer eases the burden of decode head, so that SegAdapter-augmented backbones can be equipped with a simple decoder for efficient training and inferencing without sacrificing the prediction accuracy. This makes SegAdapter a cost-effective and efficient solution to enlarge the vision backbone learning capacity instead of stacking the same base blocks. Extensive experiments demonstrate that our proposed SegAdapter can surpass the baseline and other adapter methods in a large margin, and the ablations prove that the careful designed architecture is the key component for the success of SegAdapter. For the future work, we aim to extend the SegAdapter to handle more universal segmentation tasks including instance segmentation and panoptic segmentation.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1209–1218, 2018.

Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 357–366, 2021.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022.

Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34: 17864–17875, 2021.

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

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

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.

Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. Visual attention network. *arXiv preprint arXiv:2202.09741*, 2022.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

Jitesh Jain, Anukriti Singh, Nikita Orlov, Zilong Huang, Jiachen Li, Steven Walton, and Humphrey Shi. Semask: Semantically masked transformers for semantic segmentation. *arXiv preprint arXiv:2112.12782*, 2021.

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*, pages 10012–10022, 2021.

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*, pages 11976–11986, 2022.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Yongming Rao, Wenliang Zhao, Yansong Tang, Jie Zhou, Ser Nam Lim, and Jiwen Lu. Hornet: Efficient high-order spatial interactions with recursive gated convolutions. *Advances in Neural Information Processing Systems*, 35:10353–10366, 2022.

Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5693–5703, 2019.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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.

Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders. *arXiv preprint arXiv:2301.00808*, 2023.

Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4794–4803, 2022.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34:12077–12090, 2021.

Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10819–10829, 2022.

Yuhui Yuan, Rao Fu, Lang Huang, Weihong Lin, Chao Zhang, Xilin Chen, and Jingdong Wang. Hrformer: High-resolution vision transformer for dense predict. *Advances in Neural Information Processing Systems*, 34:7281–7293, 2021.

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*, pages 633–641, 2017.