

Vision Transformer Adapter for Dense Predictions

Zhe Chen^{1,2*}, Yuchen Duan^{3*}, Wenhui Wang^{2*}, Junjun He², Tong Lu¹, Jifeng Dai², Yu Qiao²

¹Nanjing University ²Shanghai AI Laboratory ³Tsinghua University

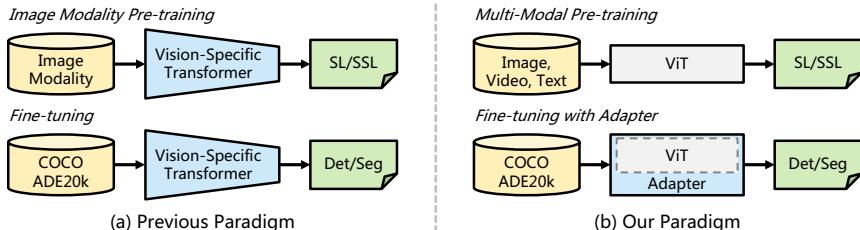


Fig. 1. Previous paradigm vs. our paradigm. (a) Previous paradigm designs vision-specific models and pre-trains on large-scale image datasets via supervised (SL) or self-supervised learning (SSL) and then fine-tunes them on downstream tasks. (b) We propose an adapter to close the gap between ViT [22] and vision-specific models on dense prediction tasks. Compared to the previous paradigm, our method preserves the flexibility of ViT and thus could benefit from advanced multi-modal pre-training.

Abstract. This work investigates a simple yet powerful adapter for Vision Transformer (ViT). Unlike recent visual transformers that introduce vision-specific inductive biases into their architectures, ViT achieves inferior performance on dense prediction tasks due to lacking prior information of images. To solve this issue, we propose a Vision Transformer Adapter (ViT-Adapter), which can remedy the defects of ViT and achieve comparable performance to vision-specific models by introducing inductive biases via an additional architecture. Specifically, the backbone in our framework is a vanilla transformer that can be pre-trained with multi-modal data. When fine-tuning on downstream tasks, a modality-specific adapter is used to introduce the data and tasks’ prior information into the model, making it suitable for these tasks. We verify the effectiveness of our ViT-Adapter on multiple downstream tasks, including object detection, instance segmentation, and semantic segmentation. Notably, when using HTC++, our ViT-Adapter-L yields 60.1 AP^b and 52.1 AP^m on COCO test-dev, surpassing Swin-L by 1.4 AP^b and 1.0 AP^m. For semantic segmentation, our ViT-Adapter-L establishes a new state-of-the-art of 60.5 mIoU on ADE20K val, 0.6 points higher than SwinV2-G. We hope that the proposed ViT-Adapter could serve as an alternative for vision-specific transformers and facilitate future research. The code and models will be released at <https://github.com/czczup/ViT-Adapter>.

Keywords: Vision Transformer, Adapter, Dense Prediction

* Equal. This work is done when Zhe Chen is an intern at Shanghai AI Lab.

1 Introduction

Recently, transformers have demonstrated remarkable success in a broad range of the computer vision field. Benefiting from the dynamic feature extraction capability and the long-term dependence of the attention mechanism, Vision Transformer (ViT) [22] and its variants [1, 11, 21, 26, 44, 52, 75] soon rose in many computer vision tasks such as object detection and semantic segmentation, surpassing CNN models and reaching state-of-the-art performance.

Although current state-of-the-art models for visual tasks are the variants of transformer introduced image prior [21, 52, 75], vanilla transformers still have some nonnegligible advantages. Stemming from the natural language processing (NLP) field, transformer has no assumption of input data. With different embedding layers, such as patch embedding [22], 3D patch embedding [53], and token embedding [73], vanilla transformers such as ViT are able to process multi-modal data in terms of image, video, and text. Therefore, ViT can use large-scale multi-modal data for pre-training [86], which makes the features captured by the model have richer semantics. However, ViT has conclusive defects in downstream tasks compared to task-specific transformers. Take the image task as an instance, lacking prior information of images results in slower convergence and lower performance, and thus vanilla transformers are not competitive with dedicated transformers [45, 75, 78] for dense prediction tasks. Inspired by the adapter [31, 66] in the NLP field, *this work aims to develop an adapter to close the gap between vanilla transformers such as ViT and the dedicated models for downstream vision tasks.*

To this end, we propose Vision Transformer Adapter (ViT-Adapter), which is an additional network that can efficiently adapt ViT to downstream dense prediction tasks without changing its original architecture. Specifically, to introduce the vision-specific inductive biases into vanilla transformers, we design three tailored modules for ViT-Adapter, including (1) a spatial prior module for capturing the local semantics (spatial prior) of input images, (2) a spatial feature injector for incorporating spatial prior into the ViT, and (3) a multi-scale feature extractor to reconstruct the multi-scale features required by dense prediction tasks.

As shown in Fig. 1, compared to the previous paradigm that pre-training on large-scale image datasets (*e.g.*, ImageNet [20]) and fine-tuning on different tasks, our

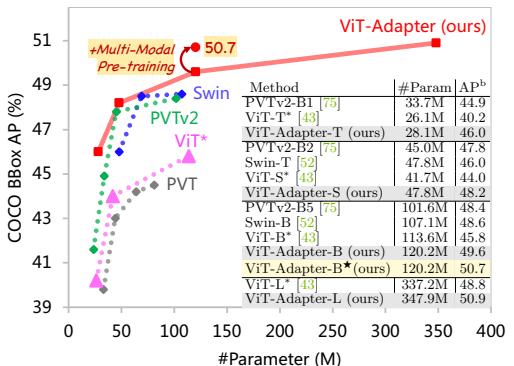


Fig. 2. Object detection performance on COCO val2017 using Mask R-CNN. We see that our method achieves significant improvements on object detection. \star indicates using multi-modal pre-trained weights from [86].

paradigm is more flexible. In our framework, the backbone network is a general model (*e.g.*, ViT) that can be pre-trained with multi-modal data and tasks. When applying it to downstream tasks, a pre-training-free vision-specific adapter is used to introduce the prior information of input data and tasks to the general backbone, making the model suitable for downstream tasks. In this way, using ViT as the backbone, our framework achieves comparable or even better performance than transformer backbones such as Swin Transformer [52], specifically designed for dense prediction tasks.

Our main contributions are as follows:

- We propose a dense prediction task adapter for ViT, that can bridge the gap between ViT and the dedicated transformers such as Swin Transformer and PVTv2 on downstream vision tasks by introducing the image prior into the ViT backbone, making it suitable for dense prediction tasks.
- To incorporate the image prior without changing the structure of ViT, we design a spatial prior module and two feature interaction operators, which can supplement the missing local continuity information of ViT and reorganize fine-grained multi-scale features for downstream tasks.
- We evaluate the proposed ViT-Adapter on multiple challenging benchmarks, including COCO [50] and ADE20K [84]. Compared to the prior arts, our models consistently achieve improved performance. As shown in Fig. 2, under comparable parameters and computation overhead, ViT-Adapter-B achieves 49.6 AP^b on COCO mini-val, outperforming Swin-B by 1.0 points (49.6 *vs.* 48.6). Benefiting from multi-modal pre-training [86], the performance of this model can be further boosted to 50.7 AP^b. When adopting HTC++ [52], our ViT-Adapter-L yields 60.1 AP^b and 52.1 AP^m on COCO test-dev. For the semantic segmentation task, our ViT-Adapter-L establishes a new state-of-the-art of 60.5 mIoU on the ADE20K dataset, 0.6 points higher than SwinV2-G [51]. We hope that this very simple and strong framework can serve as a baseline for vision-specific adapter for pure transformers.

2 Related Work

Transformers. In recent years, transformers have dominated various tasks across multiple modalities, such as natural language processing, computer vision, and speech recognition. The vanilla transformer [73] was initially proposed for machine translation and remains the state-of-the-art architecture for NLP tasks today. Vision Transformer (ViT) [22] is the first work to generalize the vanilla transformer to the image classification task without much modification. PVT [74] and Swin Transformer [52] introduce more image-specific inductive biases by incorporating the pyramid structure from CNNs, achieving superior performance in classification and dense prediction tasks while sacrificing the generalization ability of other modalities to some extent. Conformer [59] proposed the first dual network to combine CNN with transformer. Recently, BEiT [4] and MAE [27] extended the scope of ViT to self-supervised learning with masked image modeling, demonstrating the powerful potential of the pure ViT architecture.

Many works [3,36,43,86] have shown that designing modality-specific networks is an important direction, while the modality-agnostic architectures (*e.g.*, ViT) are more flexible and essential for self-supervised learning and multi-modal unified modeling. Therefore, we develop an additive adapter for ViT, which can introduce the image prior without modifying the architecture of ViT, preserving its flexibility and improving performance on dense prediction tasks.

Decoders for ViT. The architecture for dense prediction commonly follows an encoder-decoder pattern, in which the encoder generates rich features and the decoder aggregates and translates them to the final predictions. Recently, illuminated by the global receptive fields of ViT, many works employ it as the encoder and design task-specific decoders. SETR [83] is the first work to adopt ViT as the backbone and develop several CNN decoders for semantic segmentation. Segmenter [67] also extends ViT to semantic segmentation, but differs in that it equips a transformer-based decoder. DPT [61] further applies ViT to the monocular depth estimation task via a CNN decoder and yields remarkable improvements. In summary, these works improve the dense prediction performance of ViT by designing modality- and task-specific decoders, but remain ViT’s weakness of single-scale and low-resolution representation.

Adapters. To date, adapters have been widely used in the NLP field. PALS [66] and Adapters [31] introduce new modules in transformer encoders for task-specific fine-tuning, making the pre-trained model quickly adapt to downstream NLP tasks. In the field of computer vision, some adapters have been proposed for incremental learning [64] and domain adaptation [62,63]. With the advent of CLIP [60], many CLIP-based adapters [24,69,81] were presented to transfer pre-trained knowledge to zero-shot or few-shot downstream tasks. Recently, [43] employed some upsampling and downsampling modules to adapt the single-scale ViT to the multi-scale FPN [48]. This technique can be regarded as the simplest multi-scale adapter for ViT. However, its performance in dense predictions is still inferior to recent transformer variants that well combine image prior [14,15,21,75]. Therefore, it is still challenging to design a powerful adapter for improving the dense prediction performance of ViT.

3 Vision Transformer Adapter

3.1 Overall Architecture

As illustrated in Fig. 3, our model can be divided into two parts. The first part is the backbone network (*i.e.*, ViT [22]), which consists of a patch embedding followed by L transformer encoder layers (see Fig. 3(a)). The second part is the proposed ViT-Adapter as shown in Fig. 3(b), which contains (1) a spatial prior module to capture spatial features from the input image, (2) a spatial feature injector to inject spatial priors into the ViT, and (3) a multi-scale feature extractor to extract hierarchical features from the ViT.

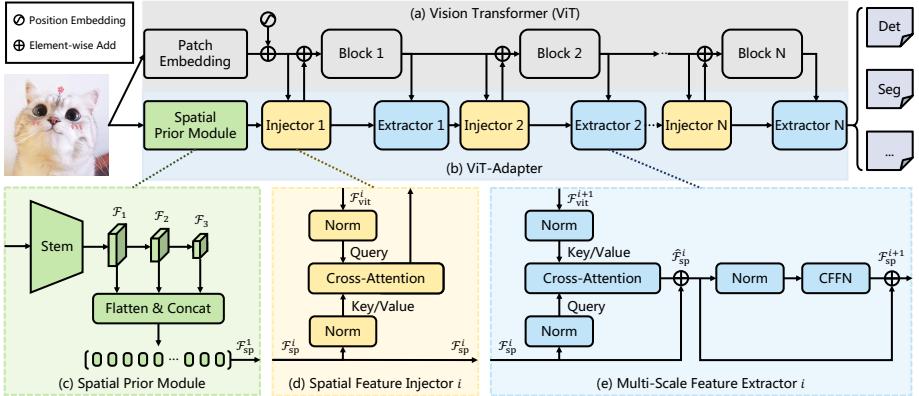


Fig. 3. Overall architecture of ViT-Adapter. (a) The Vision Transformer (ViT), whose encoder layers are divided into N equal blocks for feature interaction; (b) Our ViT-Adapter, which contains three key components; (c) The spatial prior module, which is used to model local spatial contexts from the input image; (d) The spatial feature injector for incorporating image prior into the ViT; (e) The multi-scale feature extractor for reconstructing fine-grained multi-scale features from the single-scale features of ViT.

For the ViT, the input image is first fed into the patch embedding, where the image is divided into 16×16 non-overlapping patches. After that, these patches are flattened and projected to D -dimensional embeddings. Here the feature resolution is reduced to 1/16 of the original image. Finally, the embedded patches along with the position embedding, are passed through L encoder layers of ViT.

For the ViT-Adapter, we first feed the input image into the spatial prior module. D -dimensional spatial features of three target resolutions (*i.e.*, 1/8, 1/16, and 1/32) will be collected. Then, these feature maps are flattened and concatenated as the input for feature interaction. Specifically, given the interaction times N , we evenly split the transformer encoders of ViT into N blocks, each of which contains L/N encoder layers. For the i -th block, we first inject spatial priors \mathcal{F}_{sp}^i into the block via a spatial feature injector, and then extract hierarchical features from the output of the block via a multi-scale feature extractor. After N feature interactions, we obtain high-quality multi-scale features, and then we split and reshape the features into three target resolutions 1/8, 1/16, and 1/32. Finally, we build the 1/4-scale feature map by upsampling the 1/8-scale feature map through a 2×2 transposed convolution. In this way, we obtain a feature pyramid of similar resolutions to ResNet [30], which can be used in various dense prediction tasks.

3.2 Spatial Prior Module

Recent works [75,76,78] show convolutions with overlapping sliding windows can help transformers better capture the local continuity of input images. Inspired

by this, we introduce a convolution-based *Spatial Prior Module* into ViT, which downsamples a $H \times W$ input image to different scales via a stem [30] followed by three convolutions. This module is designed to model the local spatial contexts of images parallel with the patch embedding layer, so as not to alter the original architecture of ViT.

As shown in Fig. 3(c), a standard convolutional stem borrowed from ResNet [30] is employed, which consists of three convolutions and a max-pooling layer. Next, a stack of 3×3 convolutions with the stride of 2 comprises the remainder of this module, which doubles the number of channels and reduces the size of feature maps. Finally, we adopt several 1×1 convolutions at the end to project the feature maps to D dimensions. In this way, we obtain a feature pyramid $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}$, which contains D -dimensional feature maps with resolutions of $1/8$, $1/16$, and $1/32$. Finally, we flatten and concatenate these feature maps into feature tokens $\mathcal{F}_{\text{sp}}^1 \in \mathbb{R}^{(\frac{HW}{8^2} + \frac{HW}{16^2} + \frac{HW}{32^2}) \times D}$, as the input for later feature interaction.

3.3 Feature Interaction

Due to the columnar structure, the feature maps in ViT are single-scale and low-resolution, leading to the sub-optimal performance in dense prediction tasks, compared to pyramid-structured transformers [14, 21, 52, 74, 75]. To alleviate this issue, we propose two feature interaction modules, to communicate the feature maps between our adapter and the ViT. To be specific, the two modules are based on cross-attention, namely *Spatial Feature Injector* and *Multi-Scale Feature Extractor*. As described in Section 3.1, we divide the transformer encoders of ViT to N equal blocks, and the proposed two operators are applied before and after each block, respectively.

Spatial Feature Injector. As shown in Fig. 3(d), this module is used to inject the spatial priors into ViT. Specifically, for the i -th block of the transformer, we take the input feature $\mathcal{F}_{\text{vit}}^i \in \mathbb{R}^{\frac{HW}{16^2} \times D}$ as the query, and the spatial feature $\mathcal{F}_{\text{sp}}^i \in \mathbb{R}^{(\frac{HW}{8^2} + \frac{HW}{16^2} + \frac{HW}{32^2}) \times D}$ as the key and value. We use multi-head cross-attention to inject spatial feature $\mathcal{F}_{\text{sp}}^i$ into the input feature $\mathcal{F}_{\text{vit}}^i$, which can be formulated as Eqn. 1.

$$\mathcal{F}_{\text{vit}}^i = \mathcal{F}_{\text{vit}}^i + \gamma^i \text{Attention}(\text{norm}(\mathcal{F}_{\text{vit}}^i), \text{norm}(\mathcal{F}_{\text{sp}}^i)), \quad (1)$$

where the normalization layer $\text{norm}(\cdot)$ is LayerNorm [2], and the attention layer $\text{Attention}(\cdot)$ is alternative. To reduce the computational cost, we empirically adopt deformable attention [85], a sparse attention with linear complexity, to implement the attention layer. In addition, we apply a learnable vector $\gamma^i \in \mathbb{R}^D$ to balance the attention layer's output and the input feature $\mathcal{F}_{\text{vit}}^i$, which is initialized with $\mathbf{0}$. This initialization strategy ensures that the feature distribution of $\mathcal{F}_{\text{vit}}^i$ will not be modified drastically due to the injection of spatial priors, thus making better use of the pre-trained weights of ViT.

Table 1. Configurations of the proposed ViT-Adapter. We apply our adapters on four different settings of ViT, including ViT-T, ViT-S, ViT-B, and ViT-L, which cover a wide range of different model sizes.

| Variants | Settings of ViT [22] | | | | #Param | Settings of Adapter | | | | Total Param |
|-----------|----------------------|-------|------|-------|--------|---------------------|------|-------|--------|-------------|
| | Layers | Width | FFN | Heads | | N | CFFN | Heads | #Param | |
| Tiny (T) | 12 | 192 | 768 | 3 | 5.5M | 4 | 48 | 6 | 2.5M | 8.0M |
| Small (S) | 12 | 384 | 1536 | 6 | 21.7M | 4 | 96 | 6 | 5.8M | 27.5M |
| Base (B) | 12 | 768 | 3072 | 12 | 85.8M | 4 | 192 | 12 | 14.0M | 99.8M |
| Large (L) | 24 | 1024 | 4096 | 16 | 303.3M | 4 | 256 | 16 | 23.7M | 327.0M |

Multi-Scale Feature Extractor. After injecting the spatial priors into the ViT, we obtain the output feature $\mathcal{F}_{\text{vit}}^{i+1}$ by passing $\mathcal{F}_{\text{vit}}^i$ through the encoder layers of the i -th block. After that, we swap the roles of ViT’s feature and the spatial feature. In other words, we adopt the spatial feature $\mathcal{F}_{\text{sp}}^i \in \mathbb{R}^{(\frac{HW}{8^2} + \frac{HW}{16^2} + \frac{HW}{32^2}) \times D}$ as the query, and the output feature $\mathcal{F}_{\text{vit}}^{i+1} \in \mathbb{R}^{\frac{HW}{16^2} \times D}$ as the key and value. We interact these two features via cross-attention once again, which is defined as:

$$\hat{\mathcal{F}}_{\text{sp}}^i = \mathcal{F}_{\text{sp}}^i + \text{Attention}(\text{norm}(\mathcal{F}_{\text{sp}}^i), \text{norm}(\mathcal{F}_{\text{vit}}^{i+1})). \quad (2)$$

As same as the spatial feature injector, we use deformable attention [85] here to reduce computational cost. Moreover, to remedy the defect of fixed-size position embeddings, we follow [15, 75] to introduce the convolutional feed-forward network (CFFN) after the cross-attention layer. Considering the efficiency, we refer [56] and set the ratio of CFFN to 1/4. The CFFN layer enhances the local continuity of features via depthwise convolution [13] with zero-padding, which can be expressed as:

$$\mathcal{F}_{\text{sp}}^{i+1} = \hat{\mathcal{F}}_{\text{sp}}^i + \text{CFFN}(\text{norm}(\hat{\mathcal{F}}_{\text{sp}}^i)), \quad (3)$$

where the new spatial feature $\mathcal{F}_{\text{sp}}^{i+1}$ will be used as the input of the feature interaction in the next block.

3.4 Architecture Configurations

We build our ViT-Adapter for 4 different variants of ViT [22], including ViT-T, ViT-S, ViT-B, and ViT-L. For these models, the parameter numbers of our adapters are 2.5M, 5.8M, 14.0M, and 23.7M, respectively. The details of each configuration are presented in Table 1.

In our experiments, the patch size of ViT is fixed to 16. The interaction times N is set to 4, which means we split the encoder layers of ViT into 4 equal blocks for feature interaction. Deformable attention [85] is adopted in our two feature interaction operators, in which the number of sampling points is fixed to 4, and the number of attention heads is set to 6, 6, 12, and 16. In the last interaction, we stack three multi-scale feature extractors. Besides, we set the ratio of CFFN to 1/4 to reduce the computational overhead, *i.e.* the hidden sizes of CFFN are 48, 96, 192, and 256 for 4 different ViT variants.

Table 2. Object detection and instance segmentation with Mask R-CNN on COCO val2017. “*” indicates following [43] to produce multi-scale feature maps with transposed convolutions. We initialize ViT-T/S/B with the DeiT released weights [71], and ViT-L with the weights from [65]. AP^b and AP^m represent box AP and mask AP, respectively. “MS” means multi-scale training.

| Method | #Param (M) | Mask R-CNN 1× | | | | | | Mask R-CNN 3× + MS | | | | | |
|----------------------|---------------|-----------------|-------------------------------|-------------------------------|-----------------|-------------------------------|-------------------------------|--------------------|-------------------------------|-------------------------------|-----------------|-------------------------------|-------------------------------|
| | | AP ^b | AP ^b ₅₀ | AP ^b ₇₅ | AP ^m | AP ^m ₅₀ | AP ^m ₇₅ | AP ^b | AP ^b ₅₀ | AP ^b ₇₅ | AP ^m | AP ^m ₅₀ | AP ^m ₇₅ |
| PVT-Tiny [74] | 32.9 | 36.7 | 59.2 | 39.3 | 35.1 | 56.7 | 37.3 | 39.8 | 62.2 | 43.0 | 37.4 | 59.3 | 39.9 |
| PVTv2-B0 [75] | 23.5 | 38.2 | 60.5 | 40.7 | 36.2 | 57.8 | 38.6 | 41.6 | 63.9 | 45.1 | 38.2 | 60.8 | 40.7 |
| PVTv2-B1 [75] | 33.7 | 41.2 | 61.9 | 43.9 | 25.4 | 44.5 | 54.3 | 44.9 | 67.3 | 49.4 | 40.8 | 64.0 | 43.8 |
| ViT-T* [43] | 26.1 | 36.4 | 58.9 | 38.8 | 34.2 | 55.8 | 36.1 | 40.2 | 62.9 | 43.5 | 37.0 | 59.6 | 39.0 |
| ViT-Adapter-T (ours) | 28.1 | 41.1 | 62.5 | 44.3 | 37.5 | 59.7 | 39.9 | 46.0 | 67.6 | 50.4 | 41.0 | 64.4 | 44.1 |
| PVT-Small [74] | 44.1 | 40.4 | 62.9 | 43.8 | 37.8 | 60.1 | 40.3 | 43.0 | 65.3 | 46.9 | 39.9 | 62.5 | 42.8 |
| PVTv2-B2 [75] | 45.0 | 45.3 | 67.1 | 49.6 | 41.2 | 64.2 | 44.4 | 47.8 | 69.7 | 52.6 | 43.1 | 66.8 | 46.7 |
| Swin-T [52] | 47.8 | 42.7 | 65.2 | 46.8 | 39.3 | 62.2 | 42.2 | 46.0 | 68.1 | 50.3 | 41.6 | 65.1 | 44.9 |
| Conformer-S [59] | 58.1 | 43.6 | 65.6 | 47.7 | 39.7 | 62.6 | 42.5 | - | - | - | - | - | - |
| ConvNeXt-T [54] | 48.1 | 44.2 | 66.6 | 48.3 | 40.1 | 63.3 | 42.8 | 46.2 | 67.9 | 50.8 | 41.7 | 65.0 | 44.9 |
| Focal-T [79] | 48.8 | 44.8 | 67.7 | 49.2 | 41.0 | 64.7 | 44.2 | 47.2 | 69.4 | 51.9 | 42.7 | 66.5 | 45.9 |
| UniFormer-S [40] | 41.0 | 45.6 | 68.1 | 49.7 | 41.6 | 64.8 | 45.0 | 48.2 | 70.4 | 52.5 | 43.4 | 67.1 | 47.0 |
| ViT-S* [43] | 41.7 | 40.2 | 63.1 | 43.4 | 37.1 | 59.9 | 39.3 | 44.0 | 66.9 | 47.8 | 39.9 | 63.4 | 42.2 |
| ViT-Adapter-S (ours) | 47.8 | 44.7 | 65.8 | 48.3 | 39.9 | 62.5 | 42.8 | 48.2 | 69.7 | 52.5 | 42.8 | 66.4 | 45.9 |
| PVTv2-B5 [75] | 101.6 | 47.4 | 68.6 | 51.9 | 42.5 | 65.7 | 46.0 | 48.4 | 69.2 | 52.9 | 42.9 | 66.6 | 46.2 |
| Swin-B [52] | 107.1 | 46.9 | - | - | 42.3 | - | - | 48.6 | 70.0 | 53.4 | 43.3 | 67.1 | 46.7 |
| ViT-B* [43] | 113.6 | 42.9 | 65.7 | 46.8 | 39.4 | 62.6 | 42.0 | 45.8 | 68.2 | 50.1 | 41.3 | 65.1 | 44.4 |
| ViT-Adapter-B (ours) | 120.2 | 47.0 | 68.2 | 51.4 | 41.8 | 65.1 | 44.9 | 49.6 | 70.6 | 54.0 | 43.6 | 67.7 | 46.9 |
| ViT-L* [43] | 337.2 | 45.7 | 68.9 | 49.4 | 41.5 | 65.6 | 44.6 | 48.8 | 70.9 | 53.3 | 43.6 | 67.8 | 46.8 |
| ViT-Adapter-L (ours) | 347.9 | 48.7 | 70.1 | 53.2 | 43.3 | 67.0 | 46.9 | 50.9 | 72.1 | 55.8 | 44.8 | 69.3 | 48.2 |

4 Experiments

To verify the effectiveness of our method, we conduct extensive experiments on two different dense prediction tasks, including COCO [50] object detection and instance segmentation, and ADE20K [84] semantic segmentation. We then perform ablation studies to analyze several important designs of our ViT-Adapter.

4.1 Object Detection and Instance Segmentation

Settings. Our experiments of object detection and instance segmentation are conducted on the COCO [50] benchmark, and our codes are mainly based on MMDetection [10]. We evaluate our method on top of 5 mainstream detectors, including Mask RCNN [29], Cascade Mask R-CNN [8], ATSS [82], GFL [41], and Sparse R-CNN [68]. In the training phase, we use the DeiT released weights [71] for ViT-T/S/B, and the ViT-L weights from [65] (because the DeiT release does not include ViT-L). The newly added modules of our adapter are randomly initialized and no pre-trained weights are loaded. To save time and memory, we refer [43] and modify the ViT to use 14×14 window attention in most layers. Following common practices [29, 49, 68, 74], we adopt $1 \times$ or $3 \times$ training schedule (*i.e.* 12 or 36 epochs) to train the detectors, with a batch size of 16 and AdamW [55] optimizer with an initial learning rate of 1×10^{-4} and weight decay 0.05.

Table 3. Object detection with different frameworks on COCO val2017. We initialize ViT-S with the DeiT released weights [71]. “*” indicates following [43] to produce multi-scale feature maps with transposed convolutions. “global” denotes using global attention in all layers of ViT, otherwise following the settings of [43]. AP^b represents box AP. “MS” means multi-scale training.

| Method | AP ^b | AP ₅₀ ^b | AP ₇₅ ^b | #Param | Method | AP ^b | AP ₅₀ ^b | AP ₇₅ ^b | #Param |
|--|-----------------|-------------------------------|-------------------------------|--------|-------------------------------|-----------------|-------------------------------|-------------------------------|--------|
| Cascade Mask R-CNN $3\times + \text{MS}$ | | | | | | | | | |
| Swin-T [52] | 50.5 | 69.3 | 54.9 | 86M | Swin-T [52] | 47.2 | 66.5 | 51.3 | 36M |
| Shuffle-T [35] | 50.8 | 69.6 | 55.1 | 86M | Focal-T [79] | 49.5 | 68.8 | 53.9 | 37M |
| PVTv2-B2 [75] | 51.1 | 69.8 | 55.3 | 83M | PVTv2-B2 [75] | 49.9 | 69.1 | 54.1 | 33M |
| Focal-T [79] | 51.5 | 70.6 | 55.9 | 87M | ViT-S* _{global} [43] | 46.2 | 66.0 | 50.0 | 32M |
| ViT-S* _{global} [43] | 48.4 | 67.8 | 52.1 | 82M | ViT-S* [43] | 45.2 | 64.8 | 49.0 | 32M |
| ViT-S* [43] | 47.9 | 67.1 | 51.7 | 82M | ViT-Adapter-S (ours) | 49.6 | 68.5 | 54.0 | 36M |
| ViT-Adapter-S (ours) 51.5 70.1 55.8 86M | | | | | | | | | |
| GFL $3\times + \text{MS}$ | | | | | | | | | |
| Swin-T [52] | 47.6 | 66.8 | 51.7 | 36M | Swin-T [52] | 47.9 | 67.3 | 52.3 | 110M |
| PVTv2-B2 [75] | 50.2 | 69.4 | 54.7 | 33M | Focal-T [79] | 49.0 | 69.1 | 53.2 | 111M |
| ViT-S* _{global} [43] | 47.0 | 66.4 | 50.7 | 32M | PVTv2-B2 [75] | 50.1 | 69.5 | 54.9 | 107M |
| ViT-S* [43] | 46.0 | 65.5 | 49.7 | 32M | ViT-S* _{global} [43] | 45.3 | 64.8 | 48.8 | 106M |
| ViT-Adapter-S (ours) | 50.0 | 69.1 | 54.3 | 36M | ViT-S* [43] | 44.5 | 64.3 | 48.2 | 106M |
| ViT-Adapter-S (ours) 50.0 69.1 54.3 36M | | | | | | | | | |
| Sparse R-CNN $3\times + \text{MS}$ | | | | | | | | | |
| Swin-T [52] | 47.9 | 67.3 | 52.3 | 110M | Swin-T [52] | 47.9 | 67.3 | 52.3 | 110M |
| Focal-T [79] | 49.0 | 69.1 | 53.2 | 111M | Focal-T [79] | 49.0 | 69.1 | 53.2 | 111M |
| PVTv2-B2 [75] | 50.1 | 69.5 | 54.9 | 107M | PVTv2-B2 [75] | 50.1 | 69.5 | 54.9 | 107M |
| ViT-S* _{global} [43] | 45.3 | 64.8 | 48.8 | 106M | ViT-S* _{global} [43] | 45.3 | 64.8 | 48.8 | 106M |
| ViT-S* [43] | 44.5 | 64.3 | 48.2 | 106M | ViT-S* [43] | 44.5 | 64.3 | 48.2 | 106M |
| ViT-Adapter-S (ours) | 48.1 | 67.0 | 52.4 | 110M | ViT-Adapter-S (ours) | 48.1 | 67.0 | 52.4 | 110M |
| ViT-Adapter-S (ours) 48.1 67.0 52.4 110M | | | | | | | | | |

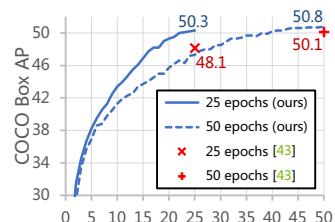
Results. As shown in Table 2, we compare our method with representative state-of-the-art backbones based on standard Mask R-CNN [29]. It shows that armed with our adapter, ViT can achieve significant improvements on object detection and instance segmentation tasks. For example, with the $3\times + \text{MS}$ training schedule, ViT-Adapter-T outperforms PVT-Tiny by 6.2 AP^b and 3.6 AP^m, while our parameter number is 15% fewer. The results of ViT-Adapter-S are 2.2 AP^b and 1.2 AP^m higher than Swin-T with similar model sizes. Moreover, our ViT-Adapter-B achieves a promising accuracy of 49.6 AP^b and 43.6 AP^m, surpassing previous state-of-the-art methods such as PVTv2-B5 and Swin-B. Our best model ViT-Adapter-L with ImageNet-22K pre-trained weights [65], yields 50.9 AP^b and 44.8 AP^m, bringing 2.1 AP^b and 1.2 AP^m gains over the baseline.

For a full comparison, we also report the performance of our method on 4 state-of-the-art detection frameworks, including Cascade R-CNN [8], ATSS [82], GFL [41], and Sparse R-CNN [68]. From Table 3, we find that our ViT-Adapter achieves comparable accuracy with specific-designed transformers [35, 52, 75, 79]. What’s more, we can see compared to the baseline that uses global attention in all layers of ViT, our adapter brings 2.8–3.4 AP^b gains.

We further compare our method with [43] and adopt the same settings. Specifically, we also use the MAE [27] pre-trained weights to initialize the ViT-B, and apply the upgraded Mask R-CNN [28] and a stronger training formula (*i.e.*, large scale jitter [25] and cosine learning rate decay). As reported in Table 4, for a schedule of 25 epochs, our ViT-Adapter-B yields higher AP^b (50.3 vs. 48.1) with only 6M additional parameters. Moreover, when training for 50 epochs, our method reaches 50.8 AP^b, which is 0.7 points better than the model without our adapter. These results also indicate that our adapter accelerates the convergence nearly twofold. In addition, we develop a heavier baseline to match our FLOPs by using global attention in all layers of ViT, which is also

Table 4. Performance comparisons with upgraded Mask R-CNN with MAE pre-trained weights. “*” indicates following [43] to produce multi-scale feature maps with transposed convolutions. “global” denotes using global attention in all layers of ViT, otherwise following the settings of [43]. AP^b and AP^m represent box AP and mask AP, respectively.

| Method | #Param | FLOPs | FPS | Epoch | AP ^b | AP ^m |
|-------------------------------|--------|-------|-----|-----------------|----------------------|----------------------|
| ViT-B* [43] | 116M | 853G | 5.1 | 25 50 100 | 48.1 50.1 50.3 | 43.2 44.6 44.9 |
| ViT-B* _{global} [43] | 116M | 1044G | 3.8 | 25 50 | 48.9 50.2 | 43.6 44.5 |
| ViT-Adapter-B (ours) | 122M | 997G | 4.4 | 25 50 | 50.3 50.8 | 44.7 45.1 |



exceeded by our method, demonstrating the powerful representation capacity of our ViT-Adapter.

4.2 Semantic Segmentation

Settings. We evaluate our ViT-Adapter on semantic segmentation with the ADE20K [84] and MMSegmentation [16]. For a complete comparison, we employ both the Semantic FPN [39] and UperNet [77] as the basic frameworks. For Semantic FPN, we apply the settings of PVT [74] and train the models for 80k iterations. For UperNet, we follow the settings of Swin [52] to train it for 160k iterations. In addition, we initialize ViT-T/S/B with the DeiT released weights [71], and ViT-L with the ImageNet-22K weights from [65].

Results. In Table 5, we report the semantic segmentation results in terms of single-scale and multi-scale mIoU. We first consider the Semantic FPN [39], which is a simple and lightweight segmentation framework without complicated designs. Under comparable model sizes, our method surpasses previous representative approaches by clear margins. For example, ViT-Adapter-T surpasses PVT-Tiny by 5.1 mIoU with nearly 30% fewer parameters. Our ViT-Adapter-S achieves 46.1 mIoU, outperforming strong counterparts such as PVTv2-B2 and Swin-T. Similarly, ViT-Adapter-B reports a competitive performance of 47.9 mIoU, which is at least 1.2 points higher than Swin-B and Twins-SVT-L. With ImageNet-22K pre-training [65], our ViT-Adapter-L yields promising accuracy of 52.9 mIoU and 53.7 multi-scale mIoU.

With the UperNet [77] framework, our method achieves generally better performance. For instance, our ViT-Adapter-T exceeds PVT-Tiny by 4.1 mIoU. Our ViT-Adapter-S yields 46.6 mIoU and 47.4 multi-scale mIoU, which is better than previous state-of-the-art transformers such as [1, 14, 52, 75, 74]. Using the ImageNet-22K pre-trained weights [65], our ViT-Adapter-B achieves 51.9 mIoU, surpassing Swin-L by 1.9 points. Furthermore, our ViT-Adapter-L yields the best performance of 53.4 mIoU and 54.4 multi-scale mIoU, which is outstanding

Table 5. Performance comparisons with different methods on the ADE20K validation set. Two different frameworks Semantic FPN [39] and UperNet [77] are used. “*” indicates following [43] to produce multi-scale feature maps with transposed convolutions. “IN-1K” and “IN-22K” represent ImageNet-1K and 22K, respectively. “MS” denotes multi-scale testing.

| Method | Pre-train | Semantic FPN 80k | | | UperNet 160k | | |
|----------------------|-----------|------------------|------|------|--------------|------|------|
| | | #Param | mIoU | +MS | #Param | mIoU | +MS |
| PVT-Tiny [74] | IN-1K | 17.0M | 36.6 | 37.3 | 43.2M | 38.5 | 39.0 |
| XCiT-T12/16 [1] | IN-1K | 8.4M | 38.1 | 39.6 | 33.7M | 41.5 | 42.2 |
| ViT-T* [43] | IN-1K | 10.2M | 39.4 | 40.5 | 34.1M | 41.7 | 42.6 |
| ViT-Adapter-T (ours) | IN-1K | 12.2M | 41.7 | 42.1 | 36.1M | 42.6 | 43.6 |
| PVT-Small [74] | IN-1K | 28.2M | 41.9 | 42.3 | 54.5M | 43.7 | 44.0 |
| PVTv2-B2 [75] | IN-1K | 29.1M | 45.2 | 45.7 | - | - | - |
| Swin-T [52] | IN-1K | 31.9M | 41.5 | - | 59.9M | 44.5 | 45.8 |
| XCiT-S12/16 [1] | IN-1K | 30.4M | 43.9 | 45.2 | 52.4M | 45.9 | 46.7 |
| Twins-PCPVT-S [14] | IN-1K | 28.4M | 44.3 | - | 54.6M | 46.2 | 47.5 |
| Twins-SVT-S [14] | IN-1K | 28.3M | 43.2 | - | 54.4M | 46.2 | 47.1 |
| ViT-S* [43] | IN-1K | 27.8M | 44.6 | 45.8 | 53.6M | 44.6 | 45.7 |
| ViT-Adapter-S (ours) | IN-1K | 31.9M | 46.1 | 46.6 | 57.6M | 46.6 | 47.4 |
| Swin-B [52] | IN-1K | 91.2M | 46.0 | - | 121.0M | 48.1 | 49.7 |
| XCiT-M24/16 [1] | IN-1K | 90.8M | 45.9 | 47.4 | 109.0M | 47.6 | 48.6 |
| Twins-SVT-L [14] | IN-1K | 103.7M | 46.7 | - | 133.0M | 48.8 | 50.2 |
| ViT-B* [43] | IN-1K | 98.0M | 46.4 | 47.6 | 127.3M | 46.1 | 47.1 |
| ViT-Adapter-B (ours) | IN-1K | 104.6M | 47.9 | 48.9 | 133.9M | 48.1 | 49.2 |
| Swin-B [52] | IN-22K | - | - | - | 121.0M | 50.0 | 51.7 |
| Swin-L [52] | IN-22K | - | - | - | 234.0M | 52.1 | 53.5 |
| ViT-Adapter-B (ours) | IN-22K | 104.6M | 50.7 | 51.9 | 133.9M | 51.9 | 52.5 |
| ViT-Adapter-L (ours) | IN-22K | 332.0M | 52.9 | 53.7 | 363.8M | 53.4 | 54.4 |

from the counterparts. Even compared to Segmenter [67], a segmentation decoder specifically designed for ViT, our method maintains an advantage of 1.6 mIoU (53.4 *vs.* 51.8). These significant and consistent improvements over different model sizes and segmentation frameworks suggest that our ViT-Adapter can cover the shortage of ViT, making it more suitable for semantic segmentation.

4.3 Ablation Study

Settings. We conduct ablation studies on the COCO [50] datasets, and initialize ViT with the weights released by DeiT [71] unless explicitly stated. All models are trained with Mask R-CNN [29] for 1× schedule and without multi-scale training. Other settings are the same as the settings in Section 4.1.

Ablation for Components. To investigate the contribution of each key design, we gradually extend the ViT-S* baseline [43] to our ViT-Adapter-S. As shown in the left side of Table 6, our spatial prior module and multi-scale feature extractor improve 3.2 AP^b and 1.6 AP^m compared to the baseline. From the results of variant 2, we find that spatial feature injector brings 0.8 AP^b and AP^m improvements, showing that the local continuity information can promote the performance of ViT on dense prediction tasks, and its extraction process can be decoupled from the architecture of ViT. Furthermore, we employ CFFN [75]

Table 6. Ablation studies of ViT-Adaptor. (Left) ablation of key components. Our proposed components collectively bring 4.5 AP^b and 2.8 AP^m gains. (Right) ablation of interaction times N . The model gives the best performance when $N = 4$. AP^b and AP^m represent box AP and mask AP, respectively. SPM is short for the spatial prior module. “#P” denotes the number of parameters.

| Method | Components | | | | Mask R-CNN 1× | | | | N | AP ^b | AP ^m | #P | |
|---------------|------------|-----|----------|-----------|---------------|-----------------|-----------------|-------|------|-----------------|-----------------|------|-------|
| | Deconv | SPM | Injector | Extractor | CFFN | AP ^b | AP ^m | #P | | | | | |
| Baseline [43] | ✓ | | | | | 40.2 | 37.1 | 41.7M | 15.4 | 0 | 40.2 | 37.1 | 41.7M |
| Variant 1 | | ✓ | | | ✓ | 43.4 | 38.7 | 45.8M | 13.1 | 1 | 43.2 | 38.9 | 45.5M |
| Variant 2 | | ✓ | ✓ | ✓ | | 44.2 | 39.5 | 47.3M | 11.9 | 2 | 43.9 | 39.4 | 46.2M |
| ViT-Adapter | ✓ | ✓ | ✓ | ✓ | ✓ | 44.7 | 39.9 | 47.8M | 11.4 | 4 | 44.7 | 39.9 | 47.8M |
| | | | | | | | | | | 6 | 44.7 | 39.8 | 49.4M |

Table 7. Comparison of different attention mechanisms. Global attention runs out of GPU memory due to its quadratic complexity. The GPU training memory is measured using V100 GPUs, with per-GPU batch size 2 and half-precision training. These results suggest that deformable attention [85] is more appropriate for our adapter. AP^b and AP^m represent box AP and mask AP, respectively.

| Attention Type | AP ^b | AP ^b ₅₀ | AP ^b ₇₅ | AP ^m | AP ^m ₅₀ | AP ^m ₇₅ | FLOPs | #Param | Memory |
|---------------------------|-----------------|-------------------------------|-------------------------------|-----------------|-------------------------------|-------------------------------|-------|--------|--------|
| Global Attention [73] | Out of Memory | | | | | | 1080G | 50.3M | N/A |
| Window Attention [73] | 42.4 | 64.2 | 46.2 | 38.5 | 61.2 | 41.2 | 466G | 50.3M | 18.8G |
| Linear SRA [75] | 41.5 | 64.0 | 44.9 | 38.2 | 61.0 | 40.9 | 410G | 51.8M | 15.0G |
| Deformable Attention [85] | 44.7 | 65.8 | 48.3 | 39.9 | 62.5 | 42.8 | 403G | 47.8M | 14.2G |

to introduce additional position information, which brings 0.5 AP^b and 0.4 AP^m gains, alleviating the drawbacks of fixed-size position embeddings used in ViT.

Interaction Times. In the right side of Table 6, we study the effect of interaction times N . Specifically, we equip ViT-S with our adapter varying different interaction times. We observe that the model accuracy saturates when the interaction times N goes larger, and applying more interactions cannot monotonically improve the performance. Therefore, we empirically set N to 4 by default.

Attention Type. In our adapter, the attention mechanism is substitutable. To verify this, we use ViT-Adapter-S as the basic model and study 4 different attention mechanisms, including global attention [73], window attention [73], linear SRA [75], and deformable attention [85]. To support handling multi-scale features, we slightly modified the window attention and linear SRA. For the window attention, we set the window sizes to 28, 14, and 7 for three different scales. As for linear SRA, we use average pooling to reduce the spatial dimension of each scale to a fixed size (*i.e.*, 7×7) before the attention operation.

The results are reported in Table 7. When adopting global attention in our adapter, it will exhaust the GPU memory (32G) due to quadratic complexity. Window attention and linear SRA can significantly reduce the computational cost, but they are limited to capturing local and global dependencies respec-

Table 8. Comparison of different pre-trained weights. It’s worth noting that our method could enjoy diverse advanced pre-training for free, such as the mask image modeling in MAE [27] and the multi-modal pre-training in Uni-Perceiver [86]. AP^b and AP^m represent box AP and mask AP, respectively. “IN-1K” denotes ImageNet-1K.

| Method | Pre-train | Data | AP ^b | AP ^b ₅₀ | AP ^b ₇₅ | AP ^m | AP ^m ₅₀ | AP ^m ₇₅ |
|---------------|--------------------|-------------|-----------------|-------------------------------|-------------------------------|-----------------|-------------------------------|-------------------------------|
| ViT-Adapter-B | DeiT [71] | IN-1K | 47.0 | 68.2 | 51.4 | 41.8 | 65.1 | 44.9 |
| | MAE [27] | IN-1K | 47.8 | 68.2 | 52.4 | 42.3 | 65.5 | 45.6 |
| | Uni-Perceiver [86] | Multi-Modal | 48.4 | 70.6 | 53.2 | 43.1 | 67.4 | 46.7 |

tively, leading to the relatively weak modeling capability. Our method avoids this problem by using deformable attention. Specifically, equipped with deformable attention, our method obtains promising performance of 44.7 AP^b and 39.9 AP^m on COCO val2017, outperforming other variants armed with window attention and linear SRA by large margins with fewer FLOPs, parameters, and GPU memory. These results indicate that deformable attention is more suitable for our adapter because of its flexibility in processing multi-scale features.

Pre-trained Weights. In this experiment, we study the effect of various pre-trained weights. For a fair comparison, we train ViT-Adapter-B using different initialization for a $1 \times$ schedule without multi-scale training. As shown in Table 8, our method can easily benefit from more advanced pre-training. For example, simply replacing the DeiT weights [71] with the MAE weights [27] gives us an extra gain of 0.8 AP^b and 0.5 AP^m. What’s more, when using the multi-modal pre-training comes from Uni-Perceiver [86], our accuracy is further boosted to 48.4 AP^b and 43.1 AP^m. These results indicate that preserving the original architecture of ViT makes our framework more flexible than specially-designed transformers, and can derive significant benefits from existing advanced pre-training approaches at no additional pre-training cost.

Feature Visualization. In Fig. 4, we visualize the multi-scale feature maps generated by ViT-B* [43] and our ViT-Adapter-B, respectively. Due to the real loss of feature resolution, the hierarchical features of ViT-B* are blurry and coarse (see Fig. 4(a)). Instead, our ViT-Adapter reconstructs fine-grained multi-scale features (see Fig. 4(b)) from the single-scale features of ViT via feature interaction, which improves localization quality and reduces missed detection.

TIDE Error Type Analysis. TIDE [5] is a toolbox for analyzing the sources of error in object detection and instance segmentation algorithms. Following [43], we show the error type analysis generated by the TIDE in Fig. 5. These results reveal more detailed information about where our method improves overall AP^b relative to the baseline [43]. For example, we observe that our ViT-Adapter slightly reduces missed errors compared to the baseline. Moreover, we see that

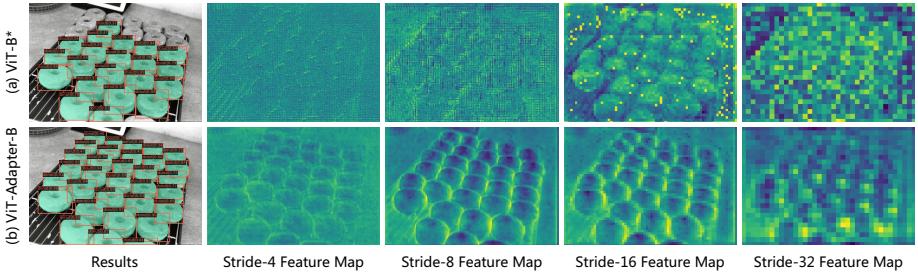


Fig. 4. Visualization of feature maps. Compared to the ViT baseline equipped with additional transposed convolutions [43], our ViT-Adapter yields fine-grained multi-scale features via multiple feature interactions, thus improving localization quality and reducing missed detection.

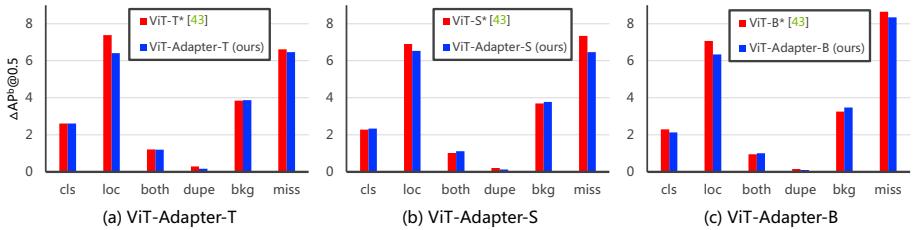


Fig. 5. TIDE error type analysis. As defined in [5], we plot the AP^b metric at an intersection-over-union (IoU) threshold of 0.5. These bars show the effect of each error type on overall detection performance. The error types include: **cls**: localized correctly but classified incorrectly; **loc**: classified correctly but localized incorrectly; **both**: classified incorrectly and localized incorrectly; **dupe**: detection would be correct if not for a higher scoring detection; **bkg**: detected background as foreground; **miss**: all undetected ground-truth not covered by other error types. We observe that our method makes fewer localization errors and miss errors than the baseline [43].

our method has a more substantial effect on fixing localization errors. This phenomenon indicates that the fine-grained hierarchical features generated by our adapter contribute to better localization quality.

5 Conclusion

This work introduces ViT-Adapter to bridge the performance gap between ViT and vision-specific transformers on dense prediction tasks. Without changing ViT’s architecture, we inject image prior into the ViT and extract the multi-scale features via a spatial prior module followed by two feature interaction operators. Extensive experiments on object detection, instance segmentation, and semantic segmentation benchmarks verify that our models can achieve comparable and even better performance than well-designed vision-specific transformers under comparable number of parameters.

Table 9. Results on COCO object detection and instance segmentation. Here we load BEiT [4] pre-trained weights to initialize our ViT-Adapter-L. AP^b and AP^m represent box AP and mask AP, respectively. “*” indicates multi-scale testing.

| Method | Framework | mini-val | | test-dev | |
|------------------------|-------------|-----------------|-----------------|-----------------|-----------------|
| | | AP ^b | AP ^m | AP ^b | AP ^m |
| Swin-L [52] | HTC++ [52] | 57.1 | 49.5 | 57.7 | 50.2 |
| Swin-L* [52] | HTC++ [52] | 58.0 | 50.4 | 58.7 | 51.1 |
| Swin-L* [52] | DyHead [19] | 58.4 | - | 58.7 | - |
| Focal-L [79] | HTC++ [52] | 57.0 | 49.9 | - | - |
| Focal-L* [79] | HTC++ [52] | 58.1 | 50.9 | 58.4 | 51.3 |
| MViTv2-L* [42] | Cascade [8] | 58.7 | 50.5 | - | - |
| CBNetV2 (Swin-L)* [46] | HTC [9] | 59.6 | 51.8 | 60.1 | 52.3 |
| ViT-Adapter-L (ours) | HTC++ [52] | 57.9 | 50.2 | 58.5 | 50.8 |
| ViT-Adapter-L* (ours) | HTC++ [52] | 59.8 | 51.7 | 60.1 | 52.1 |

Appendix

A Comparison with Previous State-of-the-Art

To further explore the potential of our method, in this section, we study whether advanced pre-training could enable ViT-Adapter-L to achieve system-level state-of-the-art performance. Because Uni-Perceiver [86] only provides the pre-trained weights for ViT-B, here we adopt the ImageNet-22K pre-trained BEiT-L [4] to initialize our ViT-Adapter-L.

BEiT [4] is a self-supervised learning approach designed for ViT [22], which proposed a masked image modeling task to learn high-quality visual representation. It achieves strong fine-tuning results on downstream tasks such as semantic segmentation, yielding state-of-the-art results on the ADE20K [84] dataset (57.0 multi-scale mIoU). Following the official repository¹ of BEiT, we also adopt the layer scale [72] and layer-wise learning rate decay to train our model. The learning rate decay rate and stochastic depth rate [32] are fixed to 0.9 and 0.3 for all experiments, respectively.

A.1 Object Detection and Instance Segmentation

Following [43], we modify the 24-layer ViT-L to use 14×14 window attention except for layers spaced at an interval of 6, to save time and memory. But due to BEiT using learnable relative position biases instead of the absolute position embeddings, we replace the remaining global attention with 56×56 window attention as an approximation. For these layers, the relative position biases need to be interpolated to accommodate the new window size.

Then, we re-implement the state-of-the-art detector HTC++ [52]. Specifically, we rescale the shorter side of the input image between 400 and 1400, while the longer side is at most 1600. Instaboost [23] augmentation, AdamW [55] optimizer (batch size of 16, initial learning rate of 1×10^{-4} , and weight decay of 0.05), and $3 \times$ schedule are adopted during training.

¹ <https://github.com/microsoft/unilm/tree/master/beit>

Table 10. Comparison with previous state-of-the-art results on the ADE20K validation set. Here we load BEiT [4] pre-trained weights to initialize our ViT-Adapter-L. “MS” denotes multi-scale testing. \dagger represents using extra data.

| Method | Framework | Crop Size | mIoU | +MS | #Param |
|--------------------------------|----------------------------|-----------|-------------|-------------|--------|
| BEiT-L [4] | UperNet [77] | 640×640 | 56.7 | 57.0 | 441M |
| BEiT-L [4] | UperNet-RR [18] | 640×640 | 57.2 | 57.7 | 441M |
| Swin-L [52] | Mask2Former [12] | 640×640 | 56.1 | 57.3 | - |
| Swin-L-FaPN [33] | Mask2Former [12] | 640×640 | 56.4 | 57.7 | - |
| SeMask-Swin-L-MSFaPN [37] | Mask2Former [12] | 640×640 | 57.0 | 58.2 | - |
| SwinV2-G \dagger [51] | UperNet [77] | 640×640 | 59.1 | - | >3.0B |
| SwinV2-G \dagger [51] | UperNet [77] | 896×896 | 59.3 | 59.9 | >3.0B |
| ViT-Adapter-L (ours) | UperNet [77] | 640×640 | 58.0 | 58.4 | 451M |
| ViT-Adapter-L (ours) | Mask2Former [12] | 640×640 | 58.3 | 59.0 | 568M |
| ViT-Adapter-L \dagger (ours) | Mask2Former \dagger [12] | 896×896 | 59.4 | 60.5 | 571M |

In Table 9, we compare the ViT-Adapter-L with previous state-of-the-art results on the COCO benchmark. Specifically, it achieves 60.1 AP^b and 52.1 AP^m on the COCO test-dev, surpassing the Swin-L [52] by 1.4 AP^b and 1.0 AP^m. This suggests that equipped with advanced pre-training and the proposed adapter, the ViT model can also yield state-of-the-art performance on the object detection and instance segmentation tasks, similar to these well-designed vision-specific transformers.

A.2 Semantic Segmentation

For semantic segmentation, we employ an AdamW [55] optimizer with an initial learning rate of 2×10^{-5} , a batch size of 16, and a weight decay of 0.05.

ADE20K [84] is a challenging dataset that includes 150 fine-grained categories. It contains 20,210 images for training and 2,000 images for validation. As shown in Table 10, when training with UperNet [77] for 160k iterations, our ViT-Adapter-L yields 58.4 multi-scale mIoU, outperforming the results of BEiT-L [4] by 1.4 points with only 10M additional parameters.

Furthermore, we adopt the more advanced Mask2Former [12] as the segmentation framework. As SwinV2-G [51] was pre-trained using a privately collected ImageNet-22K-ext-70M dataset that contains 70 million images, we additionally use the COCO-Stuff-164K [7] dataset for 80k iterations of pre-training. As the same, we fix the crop size to 896×896 pixels. It is worth noting that our ViT-Adapter-L produces a new state-of-the-art accuracy of 60.5 multi-scale mIoU under these settings, which is 0.6 points higher than the previous best model SwinV2-G [51], while the parameter number of our method is much smaller.

Cityscapes [17] is a high-resolution dataset recorded in street scenes including 19 classes. It contains 2,975 images for training, 500 images for validation, and 1,525 images for testing. In this experiment, we use Mask2Former [12] as the segmentation framework and set the crop size to 896×896 pixels. Following common practices [70,78], we first pre-train on Mapillary Vistas [58] and then

Table 11. Comparison with previous state-of-the-art results on the validation set and test set of Cityscapes. Here we load BEiT [4] pre-trained weights to initialize our ViT-Adapter-L. “MS” denotes multi-scale testing. “Coarse” represents using extra 20k coarsely annotated images.

| Method | Coarse | Crop Size | val set | | test set | |
|----------------------------------|--------|-----------|-------------|-------------|----------|-------------|
| | | | mIoU | +MS | mIoU | +MS |
| SegFormer-B5 [78] | ✗ | 1024×1024 | 82.4 | 84.0 | - | 83.1 |
| Swin-L (Mask2Former) [52] | ✗ | 512×1024 | 83.3 | 84.3 | - | - |
| SeMask-Swin-L (Mask2Former) [37] | ✗ | 512×1024 | 84.0 | 85.0 | - | - |
| Lawin-Swin-L [37] | ✗ | 1024×1024 | - | - | - | 84.4 |
| HRNet+OCR+SegFix [80] | ✓ | 769×769 | - | - | 84.5 | - |
| HRNetV2+OCR+HMS [70] | ✓ | 1024×2048 | 86.0 | 86.3 | - | 85.1 |
| ViT-Adapter-L (Mask2Former) | ✗ | 896×896 | 84.9 | 85.8 | - | 85.2 |

Table 12. Comparison with previous state-of-the-art results on COCO-Stuff-10K (left) and Pascal Context (right). Here we load BEiT [4] pre-trained weights to initialize our ViT-Adapter-L. “MS” denotes multi-scale testing.

| Method | mIoU | +MS | Method | mIoU | +MS |
|------------------------------|-------------|-------------|-----------------------------|-------------|-------------|
| ViT-L (StructToken-PWE) [47] | - | 48.2 | ViT-L (SETR-MLA) [83] | - | 55.8 |
| ViT-L (StructToken-CSE) [47] | - | 48.7 | ViT-L (Segmentor) [67] | - | 59.0 |
| ViT-L (StructToken-SSE) [47] | - | 49.1 | ViT-Hybrid (DPT) [61] | - | 60.5 |
| ConvNeXt-L+JPU (CAR) [34] | 49.0 | 50.0 | ConvNeXt-L+JPU (CAR) [34] | 63.0 | 64.1 |
| Swin-L (SenFormer) [6] | 49.8 | 51.5 | Swin-L (SenFormer) [6] | 63.1 | 64.5 |
| ViT-Adapter-L (UperNet) | 51.0 | 51.4 | ViT-Adapter-L (UperNet) | 67.0 | 67.5 |
| ViT-Adapter-L (Mask2Former) | 53.2 | 54.2 | ViT-Adapter-L (Mask2Former) | 67.8 | 68.2 |

fine-tune on Cityscapes for 80k iterations, respectively. As shown in Table 11, our ViT-Adapter-L achieves 85.2 multi-scale mIoU on the test set, which is slightly better than HRNetV2+OCR+HMS [70] that uses extra coarsely annotated data.

COCO-Stuff-10K [7] is a subset of the COCO dataset for semantic segmentation that includes 171 categories. It is split into 9,000 and 1,000 images for training and testing. Following common practices, we set the crop size to 512×512. The iteration times for UperNet and Mask2Former are 80k and 40k, respectively. As reported in Table 12 (left), our ViT-Adapter-L (Mask2Former) obtains 54.2 multi-scale mIoU, establishing a new state-of-the-art by a substantial margin of 2.7 mIoU over previous best method Swin-L (SenFormer) [6].

COCO-Stuff-164K [7] includes 118k training images with 171 categories from the COCO dataset, which is used for pre-training as marked in Table 10. This model achieves 52.0 multi-scale mIoU on the validation set of COCO-Stuff-164K.

Pascal Context [57] contains 59 semantic classes. It is divided into 4,996 images for training and 5,104 images for testing. In this experiment, the iteration times for UperNet and Mask2Former are set to 80k and 40k, with a crop size of 480×480 pixels. As shown in Table 12 (right), our ViT-Adapter-L (Mask2Former) achieves a big jump to 68.2 multi-scale mIoU.

Table 13. Object detection results on COCO-C based on Mask R-CNN. “*” indicates following [43] to produce multi-scale feature maps with transposed convolutions. ★ indecates using multi-modal pre-trained weights from [86]. The third column is the average results on 16 types of corruption data.

| Method | Clean Mean | Blur | | | | Noise | | | | Digital | | | | Weather | | | |
|----------------|-------------------------|--------|-------|-------|-------|-------|-------|------|-------|---------|-------|-------|------|---------|-------|------|-------|
| | | Motion | Defoc | Glass | Gauss | Gauss | Impul | Shot | Speck | Bright | Contr | Satur | JPEG | Snow | Spatt | Fog | Frost |
| PVTv2-B1 [75] | 44.9 35.1 | 31.1 | 30.6 | 23.7 | 33.7 | 33.0 | 31.8 | 33.2 | 36.1 | 43.3 | 42.0 | 42.5 | 30.8 | 32.4 | 40.4 | 42.1 | 34.6 |
| ViT-T* [43] | 40.2 30.0 | 26.2 | 26.5 | 22.8 | 31.2 | 27.1 | 24.7 | 27.0 | 30.5 | 38.3 | 33.8 | 37.3 | 29.3 | 25.7 | 35.1 | 35.5 | 29.1 |
| ViT-Adapter-T | 46.0 35.6 | 32.5 | 32.6 | 26.5 | 35.3 | 31.0 | 30.6 | 31.1 | 35.3 | 44.2 | 42.1 | 43.5 | 33.3 | 31.5 | 41.4 | 42.9 | 35.8 |
| PVTv2-B2 [75] | 47.8 37.7 | 34.1 | 31.5 | 27.9 | 33.9 | 36.8 | 37.0 | 37.2 | 40.0 | 46.2 | 42.5 | 40.2 | 34.1 | 35.8 | 44.0 | 43.8 | 38.3 |
| Swin-T [52] | 46.0 35.3 | 31.3 | 32.5 | 23.2 | 35.6 | 32.1 | 31.4 | 32.2 | 35.7 | 44.1 | 41.4 | 43.5 | 32.0 | 30.9 | 41.2 | 42.7 | 34.4 |
| ViT-S* [43] | 44.0 34.3 | 31.4 | 30.9 | 26.0 | 33.8 | 31.0 | 29.3 | 31.1 | 34.6 | 42.3 | 38.0 | 41.3 | 35.0 | 30.4 | 39.6 | 39.7 | 33.6 |
| ViT-Adapter-S | 48.2 37.8 | 34.5 | 34.1 | 28.9 | 37.2 | 33.2 | 33.3 | 33.5 | 37.5 | 46.4 | 43.6 | 45.8 | 37.1 | 34.2 | 43.6 | 44.7 | 37.5 |
| PVTv2-B5 [75] | 48.4 38.6 | 33.8 | 34.2 | 28.9 | 32.6 | 38.2 | 38.9 | 38.5 | 41.0 | 46.7 | 38.7 | 46.2 | 37.5 | 36.5 | 44.4 | 43.3 | 38.1 |
| Swin-B [52] | 48.6 37.8 | 33.5 | 34.3 | 25.3 | 37.7 | 35.1 | 34.7 | 35.3 | 38.6 | 46.8 | 43.7 | 46.1 | 35.8 | 33.2 | 43.9 | 45.0 | 36.5 |
| ViT-B* [43] | 45.8 36.8 | 32.6 | 33.2 | 27.9 | 36.0 | 34.6 | 33.5 | 34.7 | 37.6 | 44.3 | 41.4 | 43.5 | 36.9 | 33.1 | 41.6 | 42.3 | 35.3 |
| ViT-Adapter-B | 49.6 39.3 | 35.4 | 34.9 | 30.9 | 38.1 | 35.7 | 35.7 | 35.7 | 39.2 | 47.9 | 44.7 | 47.0 | 39.2 | 35.5 | 45.0 | 45.6 | 38.6 |
| ViT-Adapter-B★ | 50.7 43.3 | 42.2 | 37.6 | 33.0 | 40.6 | 44.3 | 39.1 | 41.0 | 43.8 | 49.5 | 49.1 | 49.5 | 42.9 | 40.5 | 48.0 | 49.4 | 42.4 |
| ViT-L* [43] | 48.8 40.2 | 35.6 | 35.8 | 30.1 | 38.8 | 38.7 | 38.0 | 38.8 | 41.1 | 46.9 | 44.7 | 46.2 | 41.0 | 38.3 | 44.9 | 45.3 | 39.1 |
| ViT-Adapter-L | 50.9 42.3 | 37.3 | 37.1 | 30.2 | 40.5 | 41.1 | 41.0 | 41.1 | 43.4 | 49.5 | 47.7 | 48.5 | 42.9 | 39.7 | 47.1 | 48.4 | 41.4 |

B Robustness to Natural Corruptions

Model robustness is important for the practical application of object detection models. In this experiment, we evaluate the robustness of ViT-Adapter to common corruptions. Specifically, we refer [38] and generate COCO-C, which expands the COCO validation set with 16 types of corruptions generated by algorithms, including blur, noise, digital, and weather categories. We compare our models with PVTv2 [75], Swin Transformer [52] and the ViT* baseline [43] based on the Mask R-CNN [29] benchmark, and calculate the mean box AP varying three severities on COCO-C.

As reported in Table 13, our models always perform better than other counterparts. It shows that the proposed ViT-Adapter not only can bring higher box AP, but also better robustness to the ViT models. What’s more, we argue that the pre-training of ViT also plays an important role in the robustness of the model. For example, using the multi-modal pre-trained weights from [86], ViT-Adapter-B★ yields an average result of 43.3 box AP on COCO-C, outperforming its normal version by 4.0 points (43.3 vs. 39.3), higher than their gap (1.1 box AP) on clean data. These results show that keeping the original architecture of ViT allows our framework to derive better robustness from existing advanced pre-training methods without additional pre-training cost.

C Feature Visualization

We provide more visualization of feature maps generated by ViT-B* [43] and our ViT-Adapter-B in Fig. 6 and Fig. 7, which are generated based on Mask R-CNN [29] and UperNet [77] frameworks respectively.

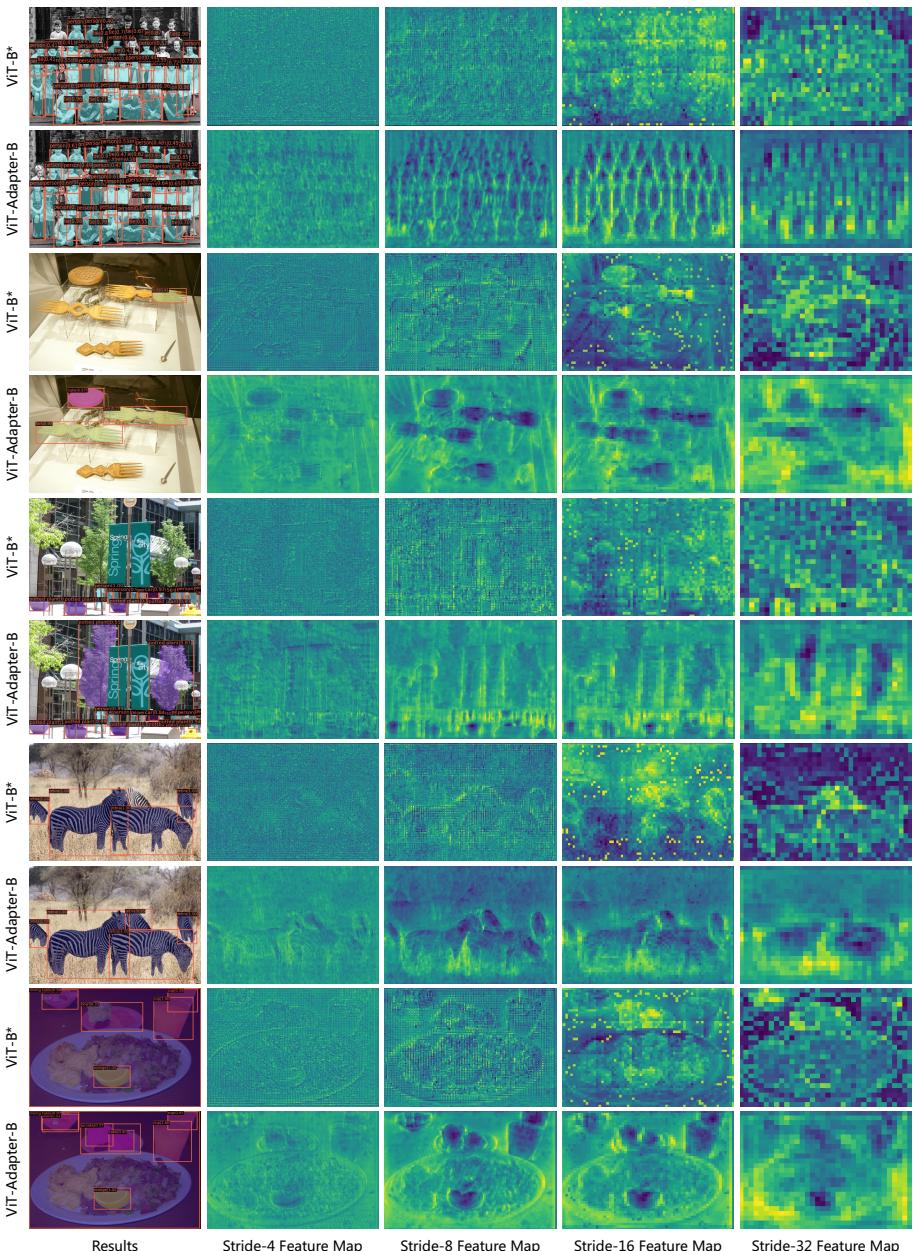


Fig. 6. Visualization of feature maps for object detection and instance segmentation. Compared to the ViT* baseline [43] equipped with additional transposed convolutions, our ViT-Adapter yields fine-grained multi-scale features via multiple feature interactions, thus improving localization quality and reducing missed detection.

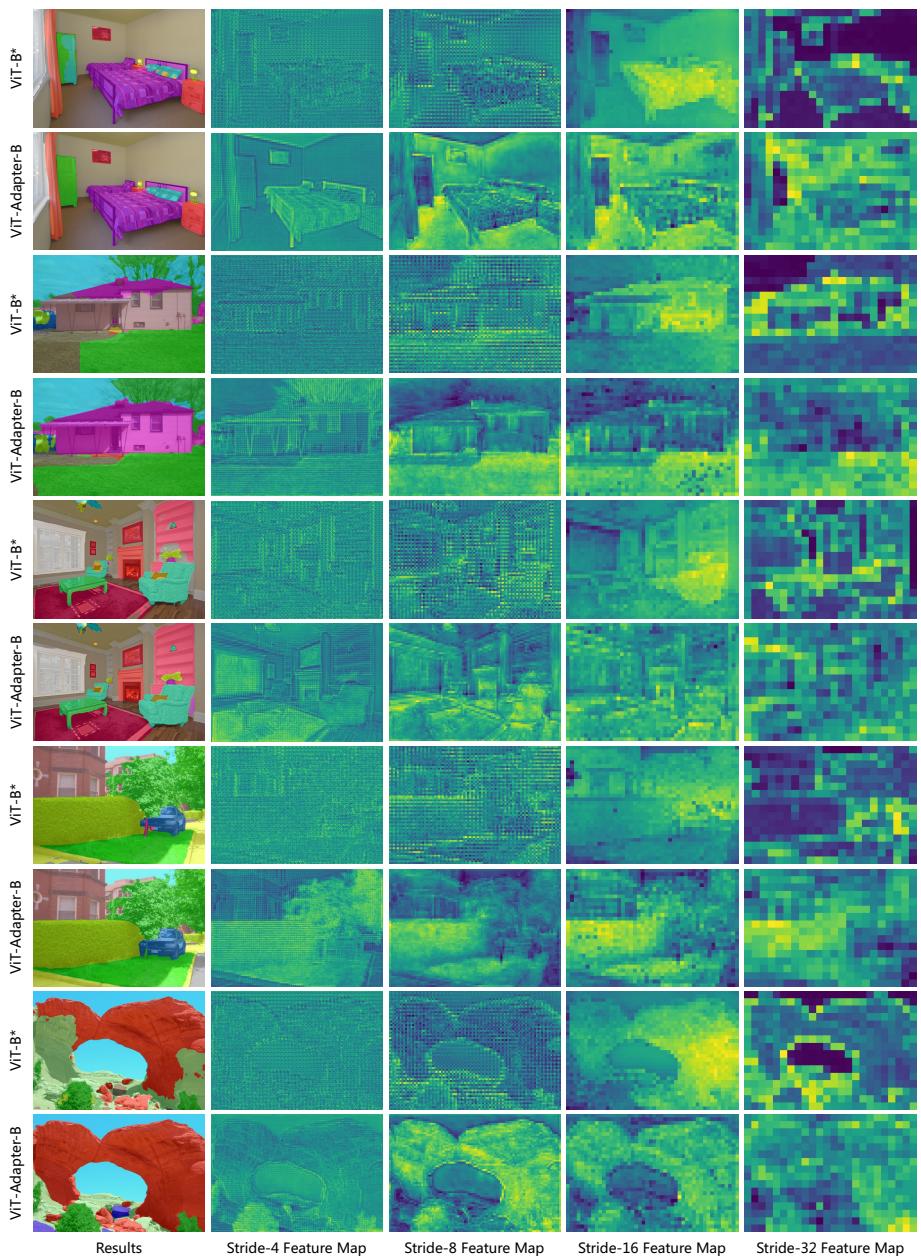


Fig. 7. Visualization of feature maps for semantic segmentation. Compared to the ViT* baseline [43] equipped with additional transposed convolutions, our ViT-Adapter yields fine-grained multi-scale features via multiple feature interactions, thus improving the performance on semantic segmentation.

References

1. Ali, A., Touvron, H., Caron, M., Bojanowski, P., Douze, M., Joulin, A., Laptev, I., Neverova, N., Synnaeve, G., Verbeek, J., et al.: Xcit: Cross-covariance image transformers. NeurIPS **34** (2021) [2](#), [10](#), [11](#)
2. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016) [6](#)
3. Baevski, A., Hsu, W.N., Xu, Q., Babu, A., Gu, J., Auli, M.: Data2vec: A general framework for self-supervised learning in speech, vision and language. arXiv preprint arXiv:2202.03555 (2022) [4](#)
4. Bao, H., Dong, L., Wei, F.: Beit: Bert pre-training of image transformers. In: ICLR (2022) [3](#), [15](#), [16](#), [17](#)
5. Bolya, D., Foley, S., Hays, J., Hoffman, J.: Tide: A general toolbox for identifying object detection errors. In: ECCV. pp. 558–573 (2020) [13](#), [14](#)
6. Bousselham, W., Thibault, G., Pagano, L., Machireddy, A., Gray, J., Chang, Y.H., Song, X.: Efficient self-ensemble framework for semantic segmentation. arXiv preprint arXiv:2111.13280 (2021) [17](#)
7. Caesar, H., Uijlings, J., Ferrari, V.: Coco-stuff: Thing and stuff classes in context. In: CVPR. pp. 1209–1218 (2018) [16](#), [17](#)
8. Cai, Z., Vasconcelos, N.: Cascade r-cnn: high quality object detection and instance segmentation. TPAMI **43**(5), 1483–1498 (2019) [8](#), [9](#), [15](#)
9. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., et al.: Hybrid task cascade for instance segmentation. In: CVPR. pp. 4974–4983 (2019) [15](#)
10. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019) [8](#)
11. Chen, W., Du, X., Yang, F., Beyer, L., Zhai, X., Lin, T.Y., Chen, H., Li, J., Song, X., Wang, Z., et al.: A simple single-scale vision transformer for object localization and instance segmentation. arXiv preprint arXiv:2112.09747 (2021) [2](#)
12. Cheng, B., Misra, I., Schwing, A.G., Kirillov, A., Girdhar, R.: Masked-attention mask transformer for universal image segmentation. arXiv preprint arXiv:2112.01527 (2021) [16](#)
13. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: CVPR. pp. 1251–1258 (2017) [7](#)
14. Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., Xia, H., Shen, C.: Twins: Revisiting the design of spatial attention in vision transformers. NeurIPS **34** (2021) [4](#), [6](#), [10](#), [11](#)
15. Chu, X., Tian, Z., Zhang, B., Wang, X., Wei, X., Xia, H., Shen, C.: Conditional positional encodings for vision transformers. arXiv preprint arXiv:2102.10882 (2021) [4](#), [7](#)
16. Contributors, M.: MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. <https://github.com/open-mmlab/mms Segmentation> (2020) [10](#)
17. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016) [16](#)
18. Cui, J., Yuan, Y., Zhong, Z., Tian, Z., Hu, H., Lin, S., Jia, J.: Region rebalance for long-tailed semantic segmentation. arXiv preprint arXiv:2204.01969 (2022) [16](#)

19. Dai, X., Chen, Y., Xiao, B., Chen, D., Liu, M., Yuan, L., Zhang, L.: Dynamic head: Unifying object detection heads with attentions. In: CVPR. pp. 7373–7382 (2021) 15
20. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255 (2009) 2
21. Dong, X., Bao, J., Chen, D., Zhang, W., Yu, N., Yuan, L., Chen, D., Guo, B.: Cswin transformer: A general vision transformer backbone with cross-shaped windows. arXiv preprint arXiv:2107.00652 (2021) 2, 4, 6
22. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2020) 1, 2, 3, 4, 7, 15
23. Fang, H.S., Sun, J., Wang, R., Gou, M., Li, Y.L., Lu, C.: Instaboost: Boosting instance segmentation via probability map guided copy-pasting. In: ICCV. pp. 682–691 (2019) 15
24. Gao, P., Geng, S., Zhang, R., Ma, T., Fang, R., Zhang, Y., Li, H., Qiao, Y.: Clip-adapter: Better vision-language models with feature adapters. arXiv preprint arXiv:2110.04544 (2021) 4
25. Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T.Y., Cubuk, E.D., Le, Q.V., Zoph, B.: Simple copy-paste is a strong data augmentation method for instance segmentation. In: CVPR. pp. 2918–2928 (2021) 9
26. Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., Wang, Y.: Transformer in transformer. NeurIPS **34** (2021) 2
27. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. arXiv preprint arXiv:2111.06377 (2021) 3, 9, 13
28. He, K., Girshick, R., Dollár, P.: Rethinking imagenet pre-training. In: ICCV. pp. 4918–4927 (2019) 9
29. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV. pp. 2961–2969 (2017) 8, 9, 11, 18
30. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016) 5, 6
31. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for nlp. In: ICML. pp. 2790–2799 (2019) 2, 4
32. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: ECCV. pp. 646–661 (2016) 15
33. Huang, S., Lu, Z., Cheng, R., He, C.: Fapn: Feature-aligned pyramid network for dense image prediction. In: ICCV. pp. 864–873 (2021) 16
34. Huang, Y., Kang, D., Chen, L., Zhe, X., Jia, W., He, X., Bao, L.: Car: Class-aware regularizations for semantic segmentation. arXiv preprint arXiv:2203.07160 (2022) 17
35. Huang, Z., Ben, Y., Luo, G., Cheng, P., Yu, G., Fu, B.: Shuffle transformer: Rethinking spatial shuffle for vision transformer. arXiv preprint arXiv:2106.03650 (2021) 9
36. Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., Carreira, J.: Perceiver: General perception with iterative attention. In: ICML. pp. 4651–4664 (2021) 4
37. Jain, J., Singh, A., Orlov, N., Huang, Z., Li, J., Walton, S., Shi, H.: Semask: Semantically masked transformers for semantic segmentation. arXiv preprint arXiv:2112.12782 (2021) 16, 17

38. Kamann, C., Rother, C.: Benchmarking the robustness of semantic segmentation models. In: CVPR. pp. 8828–8838 (2020) **18**
39. Kirillov, A., Girshick, R., He, K., Dollár, P.: Panoptic feature pyramid networks. In: CVPR. pp. 6399–6408 (2019) **10, 11**
40. Li, K., Wang, Y., Zhang, J., Gao, P., Song, G., Liu, Y., Li, H., Qiao, Y.: Uniformer: Unifying convolution and self-attention for visual recognition. arXiv preprint arXiv:2201.09450 (2022) **8**
41. Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., Tang, J., Yang, J.: Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. NeurIPS **33**, 21002–21012 (2020) **8, 9**
42. Li, Y., Wu, C.Y., Fan, H., Mangalam, K., Xiong, B., Malik, J., Feichtenhofer, C.: Improved multiscale vision transformers for classification and detection. arXiv preprint arXiv:2112.01526 (2021) **15**
43. Li, Y., Xie, S., Chen, X., Dollar, P., He, K., Girshick, R.: Benchmarking detection transfer learning with vision transformers. arXiv preprint arXiv:2111.11429 (2021) **2, 4, 8, 9, 10, 11, 12, 13, 14, 15, 18, 19, 20**
44. Li, Y., Zhang, K., Cao, J., Timofte, R., Van Gool, L.: Localvit: Bringing locality to vision transformers. arXiv preprint arXiv:2104.05707 (2021) **2**
45. Li, Z., Wang, W., Xie, E., Yu, Z., Anandkumar, A., Alvarez, J.M., Lu, T., Luo, P.: Panoptic segformer: Delving deeper into panoptic segmentation with transformers. In: CVPR (2022) **2**
46. Liang, T., Chu, X., Liu, Y., Wang, Y., Tang, Z., Chu, W., Chen, J., Ling, H.: Cbnetv2: A composite backbone network architecture for object detection. arXiv preprint arXiv:2107.00420 (2021) **15**
47. Lin, F., Liang, Z., He, J., Zheng, M., Tian, S., Chen, K.: Structtoken: Rethinking semantic segmentation with structural prior. arXiv preprint arXiv:2203.12612 (2022) **17**
48. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR. pp. 2117–2125 (2017) **4**
49. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV. pp. 2980–2988 (2017) **8**
50. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV. pp. 740–755 (2014) **3, 8, 11**
51. Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., et al.: Swin transformer v2: Scaling up capacity and resolution. arXiv preprint arXiv:2111.09883 (2021) **3, 16**
52. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV. pp. 10012–10022 (2021) **2, 3, 6, 8, 9, 10, 11, 15, 16, 17, 18**
53. Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., Hu, H.: Video swin transformer. arXiv preprint arXiv:2106.13230 (2021) **2**
54. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. arXiv preprint arXiv:2201.03545 (2022) **8**
55. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017) **8, 15, 16**
56. Mehta, S., Ghazvininejad, M., Iyer, S., Zettlemoyer, L., Hajishirzi, H.: Delight: Deep and light-weight transformer. In: ICLR (2020) **7**
57. Mottaghi, R., Chen, X., Liu, X., Cho, N.G., Lee, S.W., Fidler, S., Urtasun, R., Yuille, A.: The role of context for object detection and semantic segmentation in the wild. In: CVPR. pp. 891–898 (2014) **17**

58. Neuhold, G., Ollmann, T., Rota Bulo, S., Kortschieder, P.: The mapillary vistas dataset for semantic understanding of street scenes. In: ICCV. pp. 4990–4999 (2017) [16](#)
59. Peng, Z., Huang, W., Gu, S., Xie, L., Wang, Y., Jiao, J., Ye, Q.: Conformer: Local features coupling global representations for visual recognition. In: ICCV. pp. 367–376 (2021) [3, 8](#)
60. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML. pp. 8748–8763 (2021) [4](#)
61. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. In: ICCV. pp. 12179–12188 (2021) [4, 17](#)
62. Rebuffi, S.A., Bilen, H., Vedaldi, A.: Learning multiple visual domains with residual adapters. NeurIPS **30** (2017) [4](#)
63. Rebuffi, S.A., Bilen, H., Vedaldi, A.: Efficient parametrization of multi-domain deep neural networks. In: CVPR. pp. 8119–8127 (2018) [4](#)
64. Rosenfeld, A., Tsotsos, J.K.: Incremental learning through deep adaptation. TPAMI **42**(3), 651–663 (2018) [4](#)
65. Steiner, A., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., Beyer, L.: How to train your vit? data, augmentation, and regularization in vision transformers. arXiv preprint arXiv:2106.10270 (2021) [8, 9, 10](#)
66. Stickland, A.C., Murray, I.: Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In: ICML. pp. 5986–5995 (2019) [2, 4](#)
67. Strudel, R., Garcia, R., Laptev, I., Schmid, C.: Segmenter: Transformer for semantic segmentation. In: ICCV. pp. 7262–7272 (2021) [4, 11, 17](#)
68. Sun, P., Zhang, R., Jiang, Y., Kong, T., Xu, C., Zhan, W., Tomizuka, M., Li, L., Yuan, Z., Wang, C., et al.: Sparse r-cnn: End-to-end object detection with learnable proposals. In: CVPR. pp. 14454–14463 (2021) [8, 9](#)
69. Sung, Y.L., Cho, J., Bansal, M.: ViL-adapter: Parameter-efficient transfer learning for vision-and-language tasks. arXiv preprint arXiv:2112.06825 (2021) [4](#)
70. Tao, A., Sapra, K., Catanzaro, B.: Hierarchical multi-scale attention for semantic segmentation. arXiv preprint arXiv:2005.10821 (2020) [16, 17](#)
71. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: ICML. pp. 10347–10357 (2021) [8, 9, 10, 11, 13](#)
72. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., Jégou, H.: Going deeper with image transformers. In: ICCV. pp. 32–42 (2021) [15](#)
73. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. NeurIPS **30** (2017) [2, 3, 12](#)
74. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: ICCV. pp. 568–578 (2021) [3, 6, 8, 10, 11](#)
75. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pvttv2: Improved baselines with pyramid vision transformer. CVMJ pp. 1–10 (2022) [2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 18](#)
76. Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., Zhang, L.: CvT: Introducing convolutions to vision transformers. In: ICCV. pp. 22–31 (2021) [5](#)
77. Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J.: Unified perceptual parsing for scene understanding. In: ECCV. pp. 418–434 (2018) [10, 11, 16, 18](#)
78. Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P.: Segformer: Simple and efficient design for semantic segmentation with transformers. NeurIPS **34** (2021) [2, 5, 16, 17](#)

79. Yang, J., Li, C., Zhang, P., Dai, X., Xiao, B., Yuan, L., Gao, J.: Focal self-attention for local-global interactions in vision transformers. arXiv preprint arXiv:2107.00641 (2021) [8](#), [9](#), [15](#)
80. Yuan, Y., Chen, X., Chen, X., Wang, J.: Segmentation transformer: Object-contextual representations for semantic segmentation. In: ECCV (2020) [17](#)
81. Zhang, R., Fang, R., Gao, P., Zhang, W., Li, K., Dai, J., Qiao, Y., Li, H.: Tip-adapter: Training-free clip-adapter for better vision-language modeling. arXiv preprint arXiv:2111.03930 (2021) [4](#)
82. Zhang, S., Chi, C., Yao, Y., Lei, Z., Li, S.Z.: Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In: CVPR. pp. 9759–9768 (2020) [8](#), [9](#)
83. Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P.H., et al.: Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: CVPR. pp. 6881–6890 (2021) [4](#), [17](#)
84. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: CVPR. pp. 633–641 (2017) [3](#), [8](#), [10](#), [15](#), [16](#)
85. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. In: ICLR (2020) [6](#), [7](#), [12](#)
86. Zhu, X., Zhu, J., Li, H., Wu, X., Wang, X., Li, H., Wang, X., Dai, J.: Uni-perceiver: Pre-training unified architecture for generic perception for zero-shot and few-shot tasks. arXiv preprint arXiv:2112.01522 (2021) [2](#), [3](#), [4](#), [13](#), [15](#), [18](#)