A Simple Single-Scale Vision Transformer for Object Detection and Instance Segmentation

Wuyang Chen[1⋆], Xianzhi Du[2], Fan Yang[2]
Lucas Beyer[2], Xiaohua Zhai[2], Tsung-Yi Lin[2], Huizhong Chen[2], Jing Li[2],
Xiaodan Song[2], Zhangyang Wang[1], and Denny Zhou[2]

[1] University of Texas at Austin, Austin TX 78712, USA
{wuyang.chen,atlaswang}@utexas.edu
[2] Google
{xianzhi,fyangf,lbeyer,xzhai,tsungyi,
huizhongc,jingli,xiaodansong,dennyzhou}@google.com

Abstract. This work presents a simple vision transformer design as a strong baseline for object localization and instance segmentation tasks. Transformers recently demonstrate competitive performance in image classification. To adopt ViT to object detection and dense prediction tasks, many works inherit the multistage design from convolutional networks and highly customized ViT architectures. Behind this design, the goal is to pursue a better trade-off between computational cost and effective aggregation of multiscale global contexts. However, existing works adopt the multistage architectural design as a black-box solution without a clear understanding of its true benefits. In this paper, we comprehensively study three architecture design choices on ViT – spatial reduction, doubled channels, and multiscale features – and demonstrate that a vanilla ViT architecture can fulfill this goal without handcrafting multiscale features, maintaining the original ViT design philosophy. We further complete a scaling rule to optimize our model's trade-off on accuracy and computation cost / model size. By leveraging a constant feature resolution and hidden size throughout the encoder blocks, we propose a simple and compact ViT architecture called Universal Vision Transformer (**UViT**) that achieves strong performance on COCO object detection and instance segmentation benchmark. Our code will be available at `https://github.com/tensorflow/models/tree/master/official/projects/uvit`.

Keywords: Vision Transformer, Self-attention, Object Detection, Instance Segmentation.

1 Introduction

Transformer [49], the de-facto standard architecture for natural language processing (NLP), recently has shown promising results on computer vision tasks. Vision Transformer (ViT) [20], an architecture consisting of a sequence of transformer

⋆ Work done during the first author's research internship with Google.

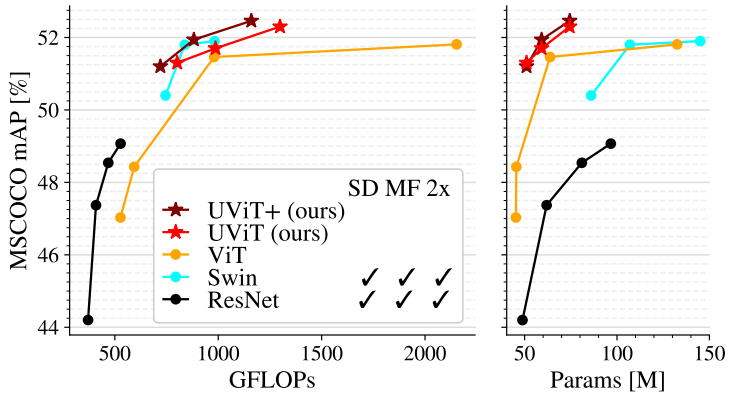

Fig. 1: Trade-off between mAP (COCO) and FLOPs (left) / number of parameters (right). We compare our UViT / UViT+ with Swin Transformer [36], ViT [55], and ResNet (18/50/101/152) [27], all adopting the same standard Cascade Mask RCNN framework [6]. Our UViT is compact, strong, and simple, avoid using any hierarchical design ("SD": spatial downsampling, "MF": multi-scale features, "2×": double channels).

encoder blocks, has achieved competitive performance compared to convolution neural networks (CNNs) [27,43] on the ImageNet classification task [19].

With the success on image classification, recent works extend transformers to more vision tasks, such as object detection [24,31,36], semantic segmentation [24,31,36,53] and video action recognition [2,17]. Conventionally, CNN architectures adopt a multistage design with gradually enlarged receptive fields [27,43] via spatial reduction or dilated convolutions. These design choices are also naturally introduced to change vision transformer architectures [28,36,53], with two main purposes: 1) **support of multi-scale features**, since dense vision tasks require the visual understanding of objects of different scales and sizes; 2) **reduction of computation cost**, for the input images of dense vision tasks are often of high resolutions, and computation complexity of vanilla self-attention is quadratic to the sequence length. The motivation behind these changes is that, tokens of the original ViT are all of a fixed fine-grained scale throughout the encoder attention blocks, which are not adaptive to dense vision applications, and more importantly incur huge computation/memory overhead. Despite the success of recent works, it is still unclear if complex design conventions of CNNs are indeed necessary for ViT to achieve competitive performance on vision tasks, and how much benefit comes from each individual design.

In this work, we demonstrate that a vanilla ViT architecture, which we call Universal Vision Transformer (**UViT**), is sufficient for achieving competitive results on the tasks of object detection and instance segmentation. Our hope is not "*to add*" any special layers to the ViT architecture (thus keeping ViT neat and simple), but instead to choose "*not to add*" complex designs that follow CNN structures (multi-scale, double channels, spatial reduction, whose compatibility with attention layers are not thoroughly verified). In other words, our goal is

not to pursue a state-of-the-art performance, but to systematically study the principles in ViT architecture designs.

First, to **support multiscale features**, instead of re-designing ViT with a multistage fashion [28,36,53], our core motivation is that self-attention mechanism naturally encourages the learning of non-local information and makes the feature pyramid no longer necessary for ViTs on dense vision tasks. This leads us to design a simple yet strong UViT architecture: we only leverage constant feature resolution and hidden size throughout the encoder blocks, and extract a single-scale feature map. Second, to **reduce the computation cost**, we adopt window splits in attention layers. We observe that on large input images for detection and instance segmentation, global attentions in early layers are redundant and compact local attentions are both effective and efficient. This motivates us to progressively increase the window sizes as the attention layers become deeper, leading to the drop of self-attention's computation cost with preserved performance.

To support the above two purposes, we systematically study fundamental architecture design choices for UViTs on dense prediction tasks. It is worth noting that, although recent works try to analyze vision transformer's generalization [38], loss landscapes [13], and patterns of learned representations [40,48], they mostly focus on image classification tasks. On dense prediction tasks like object detection and instance segmentation, many transformer models [36,28,53] directly inherit architecture principles from CNNs, without validating the actual benefit of each individual design choice. In contrast, our simple solution is based on rigorous ablation studies on dense prediction tasks, which is for the first time. Moreover, we complete a comprehensive study of UViT's compound scaling rule on dense prediction tasks, providing a series of UViT configurations that improve the performance-efficiency trade-off with highly compact architectures (even fewer than 40M parameters on transformer backbone). Our proposed UViT architectures serve as a simple yet strong baseline on COCO object detection and instance segmentation.

We summarize our contributions as below:

- We systematically study the benefits of fundamental architecture designs for ViTs on dense prediction tasks, and propose a simple UViT design that shows strong performance without hand-crafting CNN-like feature pyramid design conventions into transformers.
- We discover a new compound scaling rule (depth, width, input size) for UViTs on dense vision tasks. We find a larger input size creates more room for improvement via model scaling, and a moderate depth (number of attention blocks) outperforms shallower or deeper ones.
- We reduce the computation cost via only attention windows. We observe that attention's receptive field is limited in early layers and compact local attentions are sufficient, while only deeper layers require global attentions.
- Experiments on COCO object detection and instance segmentation demonstrate that our UViT is simple yet a strong baseline for transformers on dense prediction tasks.

2 Related Works

2.1 CNN Backbones for Dense Prediction Problems

CNNs are now mainstream and standard deep network models for dense prediction tasks in computer vision, such as object detection and semantic segmentation. During decades of development, people summarized several high-level and fundamental design conventions: 1) deeper networks for more accurate function approximation [16,21,22,32]: ResNet [27], DenseNet [30]; 2) shallow widths in early layers for high feature resolutions, and wider widths in deeper layers for compressed features, which can deliver good performance-efficiency trade-off: Vgg[43], ResNet[27]; 3) enlarged receptive fields for learning long-term correlations: dilated convolution (Deeplab series [11]), deformable convolutions [18]; 4) hierarchical feature pyramids for learning across a wide range of object scales: FPN [34], ASPP [11], HRNet [50]. In short, the motivations behind these successful design solutions fall in two folds: 1) to support the semantic understanding of objects with diverse sizes and scales; 2) to maintain a balanced computation cost under large input sizes. These two motivations, or challenges, also exist in designing our UViT architectures when we are facing dense prediction tasks, for which we provide a comprehensive study in our work (Section 3.1).

2.2 ViT Backbones for Dense Prediction Problems

The first ViT work [20] adopted a transformer encoder on coarse non-overlapping image patches for image classification and requires large-scale training datasets (JFT [44], ImageNet-21K [19]) for pretraining. DeiT further introduce strong augmentations on both data-level and architecture-level to efficiently train ViT on ImageNet-1k [19]. Beyond image classification, more and more works try to design ViT backbones for dense prediction tasks. Initially people try to directly learn high-resolution features extracted by ViT backbone via extra interpolation or convolution layers [4,58]. Some works also leverage self-attention operations to replace partial or all convolution layers in CNNs [29,41,56]. More recent trends [15,24,36,52,53,54] start following design conventions in CNNs discussed above (Section 2.1) and customize ViT architectures to be CNN-like: tokens are progressively merged to downsample the feature resolutions with reduced computation cost, along with increased embedding sizes. Multi-scale feature maps are also collected from the ViT backbone. These ViT works can successfully achieve strong or state-of-the-art performance on object detection or semantic segmentation, but the architecture is again highly customized for vision problems and lose the potential for multi-modal learning in the future. More importantly, those CNN-like design conventions are directly inherited into ViTs without a clear understanding of each individual benefit, leading to empirical black-box designs. In contrast, the simple and neat solution we will provide is motivated by a complete study on ViT's architecture preference on dense prediction tasks (Section 3.1 and 3.2).

2.3 Inductive Bias of ViT Architecture

Since the architecture of vision transformers is still in its infant stage, there are few works that systematically study principles in ViT's model design and scaling rule. Initially, people leverage coarse tokenizations, constant feature resolution, and constant hidden size [20,47], while recently fine-grained tokens, spatial downsampling, and doubled channels are also becoming popular in ViT design [36,59]. They all achieve good performance, calling for an organized study on the benefits of different fundamental designs. In addition, different learning behaviors of self-attentions (compared with CNNs) make the scaling law of ViTs highly unclear. Recent works [40] revealed ViT generates more uniform receptive fields across layers, enabling the aggregation of global information in early layers. This is contradictory to CNNs which require deeper layers to help the learning of visual global information [12]. Attention scores of ViTs are also found to gradually become indistinguishable as the encoder goes deeper, leading to identical and redundant feature maps, and plateaued performance [59]. These observations all indicate that previously discovered design conventions and scaling laws for CNNs [27,46] may not be suitable for ViTs, thus calling for comprehensive studies on the new inductive bias of ViT's architecture on dense prediction tasks.

3 Methods

Our work targets designing a simple ViT model for dense prediction tasks, and trying to avoid hand-crafted customization on architectures. We will first explain our motivations with comprehensive ablation studies on individual design benefits in Section 3.1, and then elaborate the discovered principles of our UViT designs in Section 3.2.

3.1 Is a Simple ViT Design All You Need?

As discussed in Section 1, traditionally CNNs leverage resolution downsampling, doubled channels, and hierarchical pyramid structures, to support both multi-scale features and reduction of computation cost [11,27,34]. Although recent trends in designing ViTs also inherit these techniques, it is still unclear whether they are still beneficial to ViTs. Meanwhile, ViT [20], DeiT [47], and T2T-ViT [54] demonstrate that, at least for image classification, a constant feature resolution and hidden size can also achieve competitive performance. Without spatial downsampling, the computation cost of self-attention blocks can also be reduced by using attention window splits [5,36], i.e., to limit the spatial range of the query and the key when we calculate the dot product attention scores. To better understand each individual technique and to systematically study the principles in ViT architecture designs, we provide a comprehensive study on the contributions of CNN-like design conventions to ViTs on dense prediction tasks.

Implementations: We conduct this study on the object detection task on COCO 2017 dataset. We leverage the standard Cascade Mask-RCNN detection framework [6,26], with a fixed input size as 640×640. All detection models are fine-tuned from an ImageNet pretrained initialization. More details can be found in Section 4.1.

Settings: We systematically study the benefit of spatial downsampling, multi-scale features, and doubled channels to the object detection performance of ViT. We start from a baseline ViT architecture close to the S16 model proposed in [20], which has 18 attention blocks, a hidden size of 384, and six heads for each self-attention layer. The first linear project layer embeds images into $\frac{1}{8}$-scale patches, i.e., the input feature resolution to the transformer encoder is $\frac{1}{8}$. The attention blocks will be grouped into three stages for one or a combination of two or three purposes below:

- Spatial downsampling: tokens will be merged between two consecutive stages to downsample the feature resolution. If the channel number is also doubled between stages, the tokens will be merged by a learned convolution with a stride as 2; otherwise, tokens will be merged by a 2D bi-linear interpolation.
- Multi-scale features: after each stage, features of a specific resolution will be output and fed into the detection FPN head. Multi-scale features of three target resolutions ($\frac{1}{8}, \frac{1}{16}, \frac{1}{32}$) will be collected from the encoder from early to deep attention layers.
- Doubled channels: after each of the first two stages, the token's hidden size will be doubled via a linear projection.

We study all combinations of the above three techniques, i.e. eight settings in total, and show the results in Figure 2. Note that each dot in Figure 2 indicates an individually designed and trained model. All models are of around 72 million parameters, and are trained and evaluated under the same 640×640 input size. Therefore, for vertically aligned dots, they share the same FLOPs, number of parameters, and input size, thus being fairly comparable. We control the FLOPs (x-axis) by changing the depths or attention windows allocated to different stages, see our supplement for more architecture details.

Observations

- Spatial Downsampling ("SD") does not seem to be beneficial. Our hypothesis is that, under the same FLOPs constraint, the self-attention layers already provide global features, and do not need to downsample the features to enlarge the receptive field.
- Multi-scale Features ("MF") can mitigate the poor performance from downsampling by leveraging early high-resolution features ("SD+MF"). However, the vanilla setting still outperforms this combination. We hypothesize that high-resolution features are extracted too early in the encoder; in contrast, tokens in vanilla ViTs are able to learn fine-grained details throughout the encoder blocks.

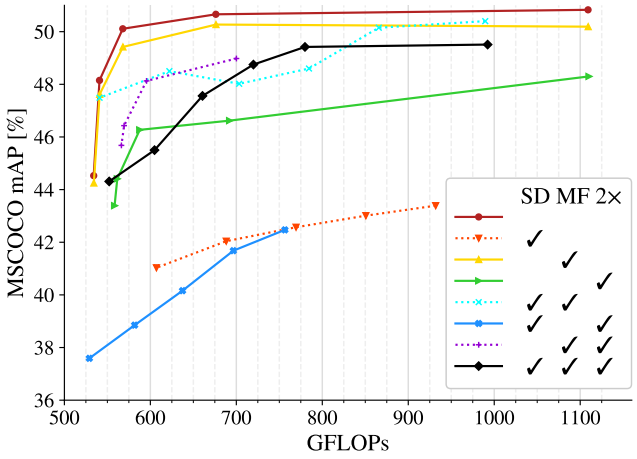

Fig. 2: The benefits of various commonly used CNN-inspired design changes to ViT: spatial downsampling ("SD"), multi-scale features ("MF"), and doubled channels ("2×"). With controlled number of parameters (72M) and input size (640×640), not using any of these designs and sticking to the original ViT model [20] performs the best in a wide range of FLOPs we explore.

– Doubled channels ("2×") plus multi-scale features ("MF") may potentially seem competitive. However, ViT does not show strong inductive bias on "deeper compressed features with more embedding dimension". This observation is also aligned with findings in [40] that ViTs have highly similar representations throughout the model, indicating that we should not sacrifice embedding dimensions of early layers to compensate for deeper layers.

In summary, we did not find strong benefits by adopting CNN-like design conventions. Instead, A simple architecture solution of a constant feature resolution and hidden size could be a strong ViT baseline.

3.2 UViT: a Simple Yet Effective Solution

Based on our study in Section 3.1, we are motivated to simplify the ViT design for dense prediction tasks and provide a neat solution, as illustrated in Figure 3. Taking 8×8 patches of input images, we learn the representation by using a constant token resolution of $\frac{1}{8}$ scale (the number of tokens remains the same) and a constant hidden size (the channel number will not be increased). A single-scale feature map will be fed into a detection or segmentation head. Meanwhile, attention windows [5] will be leveraged to reduce the computation cost.

The motivation behind our UViT design is not "*to add*" any layers to the ViT architecture, but instead to choose "*not to add*" complex designs. Our detailed study on input/model scaling will demonstrate that, a vanilla ViT architecture plus a better depth-width trade-off can achieve a high performance.

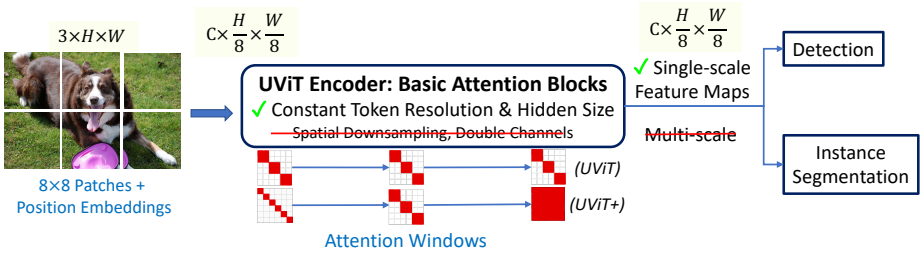

Fig. 3: We keep the architecture of our UViT neat: image patches (plus position embeddings) are processed by a stack of vanilla attention blocks with a constant resolution and hidden size. Single-scale feature maps as outputs are fed into head modules for detection or segmentation tasks. Constant (UViT, Section A.1) or progressive (UViT+, Section 3.2.2) attention windows are introduced to reduce the computation cost. We demonstrate that this simple architecture is strong, without introducing design overhead from hierarchical spatial downsampling, doubled channels, and multi-scale feature pyramids.

Though being simple, still we have two core questions to be determined in our design: (1) How to balance the UViT's depth, width, and input size to achieve the best performance-efficiency trade-off? (Section A.1) (2) Which attention window strategy can effectively save the computation cost without sacrificing the performance? (Section 3.2.2)

3.2.1 A Compound Scaling Rule of UViTs

Previous works studied compound scaling rules for CNNs [46] and ViTs [55] on image classification. However, few works studied the scaling of ViTs on dense prediction tasks. To achieve the best performance-efficiency trade-off, we systematically study the compounding scaling of UViTs on three dimensions: input size, depth, and widths. We show our results in Figure 4 and Figure 5[3]. For all models (circle markers), we first train them on ImageNet-1k, then fine-tune them on the COCO detection task.

- Depth (number of attention blocks): we study different UViT models of depths selected from $\{12, 18, 24, 32, 40\}$.
- Input size: we study three levels of input sizes: 640×640, 768×768, 896×896, and 1024×1024.
- Width (i.e. hidden size, or output dimension of attention blocks): we will tune the width to further control different model sizes and computation costs to make different scaling rules fairly comparable.

[3] This compound scaling rule is studied in Section A.1 before we study the attention window strategy in Section 3.2.2. Thus for all models in Figure 4 and Figure 5 we adopt the window scale as $\frac{1}{2}$, for fair comparisons.

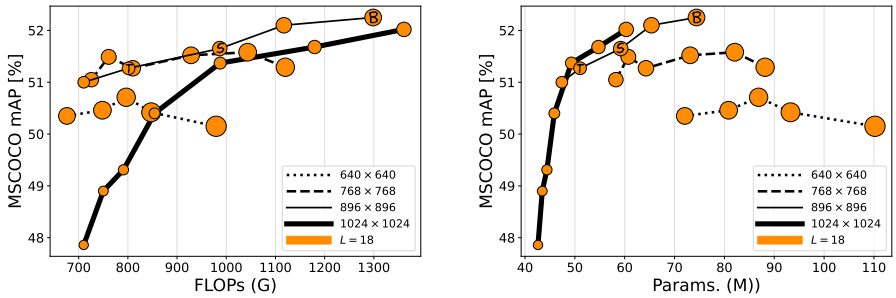

Fig. 4: **Input** scaling rule for UViT on COCO object detection. Given a fixed depth, an **input size of 896 × 896** (thin solid line) leaves more room for model scaling (by increasing the width) and is slightly better than 1024 × 1024 (thick solid line); and 640 × 640 (dashed line) or 768 × 768 (dotted line) are of worse performance-efficiency trade-off. Black capital letters "T", "S", and "B" annotate three final depth/width configurations of UViT variants we will propose (Table 2). Different sizes of markers represent the hidden sizes (widths).

Observations

- In general, UViT can achieve a strong mAP with a moderate computation cost (FLOPs) and a highly compact number of parameters (even fewer than 70M including the Cascaded FPN head).
- For input sizes (Figure 4 by different line styles): large inputs generally create more room for models to further scale up. Across a wide range of model parameters and FLOPs, we find that the scaling under an 896 × 896 input size constantly outperforms smaller input sizes (which lead to severe model overfitting), and is also better than 1024 × 1024 in a comparable FLOPs range.
- For the model depths (Figure 5), different depths in colors): we find that considering both FLOPs and the number of parameters, 18 blocks achieve better performance than 12/24/32/40 blocks. This indicates UViT needs a balanced trade-off between depth and width, instead of sacrificing depth for more width (e.g. 12 blocks) or sacrificing width for more depth (e.g. 40 blocks).

In summary, based on our final compound scaling rule, we propose our basic version of UViT as 18 attention blocks under 896 × 896 input size. See our supplement for more architecture details.

3.2.2 Attention Windows: a Progressive Strategy

In this section, we will show that a progressive attention window strategy can reduce UViT's computation cost while still preserving or even increasing the performance.

Early attentions are local operators. Originally, self-attention [49] is a global operation: unlike convolution layers that share weights to local regions, any pair of tokens in the sequence will contribute to the feature aggregation, thus collecting

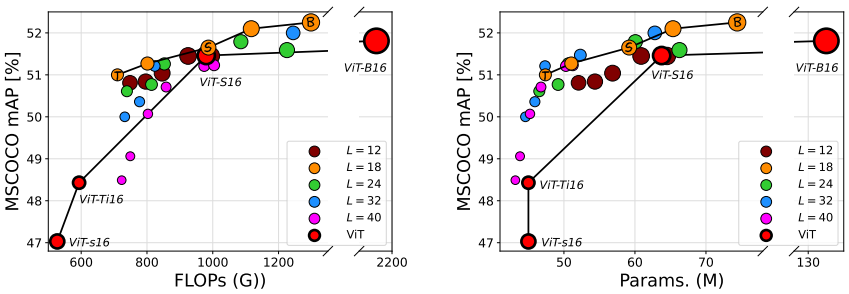

Fig. 5: **Model** scaling rule for UViT on COCO object detection. **18 attention blocks** (orange), which provide a balanced trade-off between depth and width, performs better than shallower or deeper UViTs. Black capital letters "T", "S", and "B" annotate three final depth/width configurations of UViT variants we will propose (Table 2). Different sizes of markers represent the hidden sizes (widths).

global information to each token. In practice, however, self-attention in different layers may still have biases in regions they prefer to focus on.

To validate this assumption, we select a pretrained ViT-B16 model [20], and calculate the relative receptive field of each self-attention layer on COCO. Given a sequence feature of length L and the attention score $\boldsymbol{s}$ (after softmax) from a specific head, the relative receptive field r is defined as:

$$r = \frac{1}{L}\sum_{i=1}^{L}\frac{\sum_{j=1}^{L}\boldsymbol{s}_{i,j}|i-j|}{\max(i, L-i)}, i, j = 1, \cdots, L, \tag{1}$$

where $\sum_{j=1}^{L}\boldsymbol{s}_{i,j} = 1$ for $j = 1, \cdots, L$. This relative receptive filed takes into consideration the token's position and the furthest possible location a token can aggregate, and indicates the spatial focus of the self-attention layer.

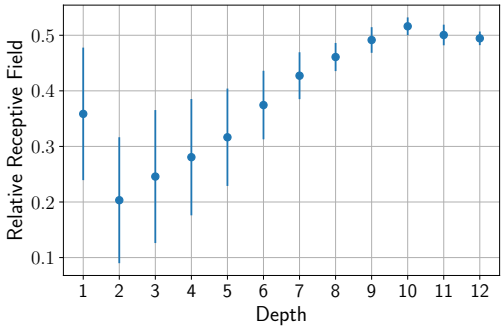

Fig. 6: Relative attention's receptive field of a ImageNet pretrained ViT-B16 [20] along depth (indices of attention blocks), on the COCO dataset. Error bars are standard deviations across different attention heads.

We collect the averages and standard deviations across different attention heads. As shown in Figure 6, we can see that tokens in early attention layers,

although having the potential to aggregate long-range features, weight more on their neighbor tokens and thus act like a "local operator". As the attention layers stack deeper, the receptive field increases, transiting the self-attention to a global operation. This inspires us that, if we explicitly limit the attention range of early layers, we may save the computation cost but still preserve the capability of self-attentions.

Attention window improves UViT's efficiency. Motivated by Figure 6, we want to study the most effective attention window strategy. Specifically, we need principled answers to two design questions:

1) *How small a window size can early attention layers endure?* To answer this question, we start the attention blocks with square windows with different small scales: $\{\frac{1}{16}, \frac{1}{8}, \frac{1}{4}\}$ of height or width a sequence's 2D shape[4].
2) *Do deeper layers require global attention, or some local attentions are also sufficient?* To compare with global attentions (window size as 1), we will also try small attention windows (window size of $\frac{1}{2}$ scale) in deeper layers.

To represent an attention window strategy that "progressively increases window scales from $\frac{1}{4}$ to $\frac{1}{2}$ to 1", we use a simple annotation "$[4^{-1}]\times 14 \rightarrow [2^{-1}]\times 2 \rightarrow [1]\times 2$", indicating that there are 14 attention blocks assigned with $\frac{1}{4}$-scale windows, then two attention blocks assigned with $\frac{1}{2}$-scale windows, and finally two attention blocks assigned with 1-scale windows. When comparing different window strategies, we make sure all strategies have the same number of parameters and share the similar computation cost for fair comparisons. We also include four more baselines of constant attention window scale across all attention blocks: global attention, and also windows of $\frac{1}{4} \sim \frac{1}{2}$ scale. We show our results in Table 1, and summarize observations below:

- With smaller constant window scale ($\frac{1}{2}/\frac{1}{3}/\frac{1}{4}$), we save more computation cost with slight sacrifice in mAP.
- Adopting constant global attentions throughout the whole encoder blocks (window size as 1, first row) is largely redundant, which contributes marginal benefits but suffers from a huge computation cost.
- Early attentions can use smaller windows like $\frac{1}{4}$-scale, but over-shrank window sizes ($\frac{1}{16}, \frac{1}{8}$) can impair the capability of self-attentions (3rd, 4th rows).
- Deeper layers still require global attentions to preserve the final performance (last two rows).
- A properly designed window strategy (5th row) can outperform vanilla solutions (1st, 2nd row) with a reduced computation cost.

In conclusion, we set the window scale of our basic version (UViT, Section A.1) as constant 2^{-1}, and proposed an improved version of our model, dubbed "UViT+" with the attention window strategy adopted as "$[4^{-1}]\times 14 \rightarrow [2^{-1}]\times 2 \rightarrow [1]\times 2$".

[4] For example, if the input sequence has $(896/8) \times (896/8) = 112 \times 112$ tokens, a window of scale $\frac{1}{16}$ will contain $7 \times 7 = 49$ elements. Similar ideas for $\frac{1}{8}$ and $\frac{1}{4}$.

Table 1: Over-shrank window sizes in early layers are harmful, and global attention windows in deep layers are vital to the final performance. Fractions in brackets indicate attention window scales (relative to sequence feature sizes), and the multiplier indicates the number of attention blocks allocated to an attention window scale (18 blocks in total). Standard Deviations of three random runs are shown in parentheses.

$[\text{window\_scale}] \times \#\text{layers}$	GFLOPs	AP_{val}	Img/s
$[1] \times 18$	2961.9	52.4 (0.09)	3.5
$[2^{-1}] \times 18$	1298.7	52.3 (0.17)	10.5
$[16^{-1}] \times 4 \rightarrow [8^{-1}] \times 4 \rightarrow [4^{-1}] \times 4 \rightarrow [2^{-1}] \times 4 \rightarrow [1] \times 2$	1154.3	52.0 (0.15)	11.5
$[8^{-1}] \times 9 \rightarrow [4^{-1}] \times 4 \rightarrow [2^{-1}] \times 3 \rightarrow [1] \times 2$	1131.2	52.2 (0.21)	12.7
$[4^{-1}] \times 14 \rightarrow [2^{-1}] \times 2 \rightarrow [1] \times 2$	1160.1	52.5 (0.11)	12.3
$[4^{-1}] \times 6 \rightarrow [2^{-1}] \times 12$	1160.1	52.2 (0.12)	12.5

4 Final Results

We conduct our experiments on COCO [35] object detection and instance segmentation to show our final performance.

4.1 Implementations

We implement our model and training in TensorFlow and Keras. Experiments are conducted on TPUs. Before fine-tuning on object detection or instance segmentation, we follow the DeiT [47] training settings to pretrain our UViTs on ImageNet-1k with a 224×224 input size and a batch size of 1024. We follow the convention in [20]: during ImageNet pretraining the kernel size of the first linear projection layer is 16×16. During fine-tuning, we will use a more fine-grained 8×8 patch size for the dense sampling purpose. The kernel weight of the first linear project layer will be interpolated from 16×16 to 8×8, and the position embedding will also be elongated by interpolation. We also report the throughput ("Img/s"), which measures the latency of UViTs by feeding one image per TPU core.

4.2 Architectures and ImageNet Pretraining

We propose three variants of our UViT variants. The architecture configurations of our model variants are listed in Table 2, and are also annotated in Figure 4 and Figure 5 ("T", "S", "B" in black). The number of heads is fixed as six, and the expansion ratio of each FFN (feed-forward network) layer if fixed as four in all experiments. As discussed in Section 3.2.2, the attention window strategy will be "$[4^{-1}] \times 14 \rightarrow [2^{-1}] \times 2 \rightarrow [1] \times 2$".

Table 2: Architecture variants of our UViT with ImageNet [19] Pretraining Performance.

Name	Depth	Hidden Size	Params. (M)	GFLOPs	Top-1	Img/s
UViT-T	18	222	13.5	2.5	76.0%	170.2
UViT-S	18	288	21.7	4.0	78.9%	145.4
UViT-B	18	384	32.8	6.9	81.3%	134.2

Table 3: Two-stage object detection and instance segmentation results on COCO 2017. We compare employing different backbones with Cascade Mask R-CNN on single model without test-time augmentation. UViT sets a constant window scale as 2^{-1}, and UViT+ adopts the attention window strategy as "$[4^{-1}] \times 14 \rightarrow [2^{-1}] \times 2 \rightarrow [1] \times 2$". We also reproduced the performance of ResNet under the same settings.

Backbone	Resolution	GFLOPs	Params. (M)	AP_{val}	AP^{mask}_{val}	Img/s
ResNet-18	896×896	370.4	48.9	44.2	38.5	-
ResNet-50	896×896	408.8	61.9	47.4	40.8	-
Swin-T [36]	480~800×1333	745	86	50.5	43.7	15.3
Shuffle-T [31]	480~800×1333	746	86	50.8	44.1	-
UViT-T (ours)	896×896	801.4	51.0	**51.3**	43.6	11.8
UViT-T+ (ours)	896×896	720.2	51.0	**51.2**	**43.9**	14.2
ResNet-101	896×896	468.2	81.0	48.5	41.8	-
Swin-S [36]	480~800×1333	838	107	51.8	44.7	12.0
Shuffle-S [31]	480~800×1333	844	107	**51.9**	**44.9**	-
UViT-S (ours)	896×896	986.8	59.2	51.7	44.1	11.1
UViT-S+ (ours)	896×896	882.2	59.2	**51.9**	44.5	12.5
ResNet-152	896×896	527.7	96.7	49.1	42.1	-
Swin-B [36]	480~800×1333	982	145	51.9	45	11.6
Shuffle-B [31]	480~800×1333	989	145	52.2	45.3	-
GCNet [7]	-	1041	-	51.8	44.7	-
UViT-B (ours)	896×896	1298.7	74.4	52.3	44.3	10.5
UViT-B+ (ours)	896×896	1160.1	74.4	**52.5**	44.8	12.3
UViT-B+ (ours) w/ self-training	896×896	1160.1	74.4	**53.9**	**46.1**	12.3

4.3 COCO detection & instance segmentation

Settings Object detection experiments are conducted on COCO 2017 [35], which contains 118K training and 5K validation images. We consider the popular Cascade Mask-RCNN detection framework [6,26], and leverage multi-scale training [45,8] (resizing the input to 896×896), AdamW optimizer [37] (with an initial learning rate as 3×10^{-3}), weight decay as 1×10^{-4}, and a batch size of 256. Similar above, the thoughput ("Img/s") measures the latency of UViTs with one COCO image per TPU core.

From Table 3 we can see that on different levels of model variants, our UViTs are highly compact. Compared with both CNNs and other ViT works, our UViT achieves strong results with much better efficiency: with similar GFLOPs, UViT uses a much fewer number of parameters (at least **44.9%** parameter reduction compared with Swin [36]). To make this comparison clean, we did not adopt any system-level techniques [36] to boost the performance[5]. As we did not leverage any CNN-like hierarchical pyramid structures, the results of our simple and neat solution suggest that, the original design philosophy of ViT [20] is a strong baseline without any hand-crafted architecture customization. We also show the mAP-efficiency trade-off curve in Figure 1. Besides, we also adopt our UViT-B backbone with the Mask-RCNN [26] framework, and achieve 50.5 AP_{val} with 1026.1 GFLOPs.

Additionally, we adopt self-training on top of our largest model (UViT-B) to evaluate the performance gain by leveraging unlabeled data, similar as [60]. We use ImageNet-1K without labels as the unlabeled set, and a pretrained UViT-B model as the teacher model to generate pseudo-labels. All predicted boxes with confidence scores larger than 0.5 are kept, together with their corresponding masks. For UViT-B with self-training, the student model is initialized from the same weights of the teacher model. The ratio of labeled data to pseudo-labeled data is 1:1 in each batch. Apart from increasing training steps by 2× for each epoch, all other hyperparameters remain unchanged. We can see from the last row in Table 3 that self-training significantly improves box AP and mask AP by 1.4% and 1.3%, respectively.

5 Conclusion

We present a simple, single-scale vision transformer backbone that can serve as a strong baseline for object detection and semantic segmentation. Our novelty is not "to add" any special layers to ViT, but instead to choose "not to add" complex designs, with strong motivations and clear experimental supports. ViT is proposed for image classification. To adapt ViT to dense vision tasks, recent works choose "to add" more CNN-like designs (multi-scale, double channels, spatial reduction). But these add-ons mainly follow the success of CNNs, and their compatibility with attention layers is not verified. However, our detailed study shows that CNN-like designs are not prerequisites for ViT, and a vanilla ViT architecture plus a better scaling rule (depth, width, input size) and a progressive attention widow strategy can indeed achieve a high detection performance. Our proposed UViT architectures achieve strong performance on both COCO object detection and instance segmentation. Our uniform design has the potential of supporting multi-modal/multi-task learning and vision-language problems. Most importantly, we hope our work could bring the attention to the community that ViTs may require careful and special architecture design on dense prediction tasks, instead of directly adopting CNN design conventions in black-box.

[5] As we adopt the popular Cascade Mask-RCNN detection framework [6,26], some previous detection works [14,45] may not be directly compared.

References

1. Amirul Islam, M., Rochan, M., Bruce, N.D., Wang, Y.: Gated feedback refinement network for dense image labeling. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3751–3759 (2017)
2. Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lucic, M., Schmid, C.: Vivit: A video vision transformer. ArXiv **abs/2103.15691** (2021)
3. Artacho, B., Savakis, A.: Waterfall atrous spatial pooling architecture for efficient semantic segmentation. Sensors **19**(24), 5361 (2019)
4. Beal, J., Kim, E., Tzeng, E., Park, D.H., Zhai, A., Kislyuk, D.: Toward transformer-based object detection. arXiv preprint arXiv:2012.09958 (2020)
5. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150 (2020)
6. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6154–6162 (2018)
7. Cao, Y., Xu, J., Lin, S., Wei, F., Hu, H.: Global context networks. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020)
8. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision. pp. 213–229. Springer (2020)
9. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062 (2014)
10. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence **40**(4), 834–848 (2017)
11. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
12. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV). pp. 801–818 (2018)
13. Chen, X., Hsieh, C.J., Gong, B.: When vision transformers outperform resnets without pretraining or strong data augmentations. arXiv preprint arXiv:2106.01548 (2021)
14. Chen, Y., Zhang, Z., Cao, Y., Wang, L., Lin, S., Hu, H.: Reppoints v2: Verification meets regression for object detection. Advances in Neural Information Processing Systems **33**, 5621–5631 (2020)
15. Chu, X., Zhang, B., Tian, Z., Wei, X., Xia, H.: Do we really need explicit position encodings for vision transformers? arXiv e-prints pp. arXiv–2102 (2021)
16. Cohen, N., Sharir, O., Shashua, A.: On the expressive power of deep learning: A tensor analysis. In: Conference on learning theory. pp. 698–728. PMLR (2016)
17. Crotts, A.P.S.: Vatt/columbia microlensing survey of m31 and the galaxy. arXiv: Astrophysics (1996)
18. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 764–773 (2017)
19. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. IEEE (2009)

20. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
21. Elbrächter, D., Perekrestenko, D., Grohs, P., Bölcskei, H.: Deep neural network approximation theory. arXiv preprint arXiv:1901.02220 (2019)
22. Eldan, R., Shamir, O.: The power of depth for feedforward neural networks. In: Conference on learning theory. pp. 907–940. PMLR (2016)
23. Ghiasi, G., Fowlkes, C.C.: Laplacian pyramid reconstruction and refinement for semantic segmentation. In: European conference on computer vision. pp. 519–534. Springer (2016)
24. Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., Wang, Y.: Transformer in transformer. arXiv preprint arXiv:2103.00112 (2021)
25. Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: 2011 International Conference on Computer Vision. pp. 991–998. IEEE (2011)
26. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
27. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
28. Heo, B., Yun, S., Han, D., Chun, S., Choe, J., Oh, S.J.: Rethinking spatial dimensions of vision transformers. arXiv preprint arXiv:2103.16302 (2021)
29. Hu, H., Zhang, Z., Xie, Z., Lin, S.: Local relation networks for image recognition. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3464–3473 (2019)
30. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
31. 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)
32. Liang, S., Srikant, R.: Why deep neural networks for function approximation? arXiv preprint arXiv:1610.04161 (2016)
33. Lin, G., Milan, A., Shen, C., Reid, I.: Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1925–1934 (2017)
34. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
35. 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: European conference on computer vision. pp. 740–755. Springer (2014)
36. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. arXiv preprint arXiv:2103.14030 (2021)
37. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
38. Naseer, M., Ranasinghe, K., Khan, S., Hayat, M., Khan, F.S., Yang, M.H.: Intriguing properties of vision transformers. arXiv preprint arXiv:2105.10497 (2021)

39. Peng, C., Zhang, X., Yu, G., Luo, G., Sun, J.: Large kernel matters–improve semantic segmentation by global convolutional network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4353–4361 (2017)
40. Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., Dosovitskiy, A.: Do vision transformers see like convolutional neural networks? arXiv preprint arXiv:2108.08810 (2021)
41. Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., Shlens, J.: Stand-alone self-attention in vision models. arXiv preprint arXiv:1906.05909 (2019)
42. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
43. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
44. Sun, C., Shrivastava, A., Singh, S., Gupta, A.: Revisiting unreasonable effectiveness of data in deep learning era. In: Proceedings of the IEEE international conference on computer vision. pp. 843–852 (2017)
45. 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: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14454–14463 (2021)
46. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946 (2019)
47. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. arXiv preprint arXiv:2012.12877 (2020)
48. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., Jégou, H.: Going deeper with image transformers. arXiv preprint arXiv:2103.17239 (2021)
49. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30**, 5998–6008 (2017)
50. Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al.: Deep high-resolution representation learning for visual recognition. IEEE transactions on pattern analysis and machine intelligence (2020)
51. Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., Cottrell, G.: Understanding convolution for semantic segmentation. In: 2018 IEEE winter conference on applications of computer vision (WACV). pp. 1451–1460. IEEE (2018)
52. 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. arXiv preprint arXiv:2102.12122 (2021)
53. Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P.: Segformer: Simple and efficient design for semantic segmentation with transformers. arXiv preprint arXiv:2105.15203 (2021)
54. Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Tay, F.E., Feng, J., Yan, S.: Tokens-to-token vit: Training vision transformers from scratch on imagenet. arXiv preprint arXiv:2101.11986 (2021)
55. Zhai, X., Kolesnikov, A., Houlsby, N., Beyer, L.: Scaling vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12104–12113 (2022)
56. Zhao, H., Jia, J., Koltun, V.: Exploring self-attention for image recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10076–10085 (2020)

57. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2881–2890 (2017)
58. 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: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6881–6890 (2021)
59. Zhou, D., Kang, B., Jin, X., Yang, L., Lian, X., Jiang, Z., Hou, Q., Feng, J.: Deepvit: Towards deeper vision transformer. arXiv preprint arXiv:2103.11886 (2021)
60. Zoph, B., Ghiasi, G., Lin, T.Y., Cui, Y., Liu, H., Cubuk, E.D., Le, Q.V.: Rethinking pre-training and self-training. arXiv preprint arXiv:2006.06882 (2020)

A Pascal VOC semantic segmentation

Settings Semantic segmentation experiments are conducted on Pascal VOC 2012, which contains 20 foreground classes and 1 background. For training, we use an augmented version of the dataset [25] with extra annotations of 10582 images (trainaug). The default training setup uses scale jittering of [0.5, 2.0] and random horizontal image flipping.

A.1 Scaling rule of UViTs

To also achieve the best performance-efficiency trade-off on semantic segmentation task, we further systematically study the model scaling of UViTs on depths and widths on the Pascal VOC dataset. We show our results[6] in Figure 7. For all models (circle markers), we first train them on ImageNet-1k, then directly fine-tune them on Pascal VOC. We fix the input size as 512×512.

- Depth (number of attention blocks): we study different UViT models of depths selected from $\{12, 18, 24, 32\}$.
- Width (i.e. hidden size, or output dimension of attention blocks): we will tune the width to further control different model sizes and computation costs to make different scaling rules fairly comparable.

In summary, based on our compound scaling rule, we find the UViT of depth 32 performs the best on Pascal VOC.

A.2 Architectures

We propose three variants of our UViT variants. The architecture configurations of our model variants are listed in Table 4, and are also annotated in Figure 7 ("T", "S", "B" in white). The number of heads is fixed as six, and the expansion ratio of each FFN (feed-forward network) layer is fixed as four in all experiments. We also scale up our UViT into a huge version following the design in [20,55], and denote it as "UViT-H".

[6] This scaling rule is studied before we study the attention window strategy in Section A.3. Thus for all models in Figure 7 we adopt the window scale as $\frac{1}{2}$, for fair comparisons.

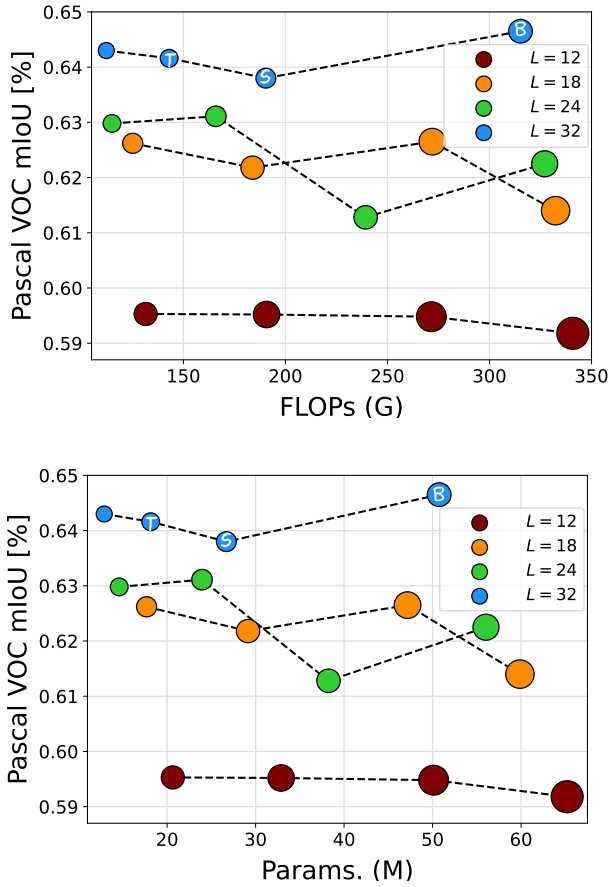

Fig. 7: **Model** scaling rule for UViT on Pascal VOC semantic segmentation (ImageNet pretrained, before COCO pretraining). **32 attention blocks** (blue) perform better than shallower UViTs. Different sizes of markers represent the hidden sizes (widths).

Table 4: Architecture variants of our UViT for Pascal VOC semantic segmentation.

Name	Depth	Hidden Size	Params. (M)
UViT-T	32	192	18.1
UViT-S	32	240	26.7
UViT-B	32	342	50.7
UViT-H	32	1280	529.9

A.3 Attention windows on Pascal VOC

In this section, we further study the attention window strategy on the Pascal VOC dataset, using our UViT-B. As shown in Table 5, progressive attention

windows again achieve the best performance. Global attentions in deep layers are vital, and a smaller window in early attentions can improve efficiency. In conclusion, we set the window scale of our UViT as "$[2^{-1}] \times 28 \rightarrow [1^{-1}] \times 4$" for its reduced computation cost.

Table 5: Local attention windows in early layers can improve model efficiency, and global attention windows in deep layers are vital to the final performance on Pascal VOC. Model: UViT-B.

$[\text{window\_scale}] \times$ #layers	GFLOPs	AP_{val}
$[1] \times 32$	596.7	81.1
$[2^{-1}] \times 32$	315.2	80.6
$[2^{-1}] \times 28 \rightarrow [1^{-1}] \times 4$	350.4	81.2

A.4 Final performance on Pascal VOC

Following the same procedure in [9,42,23,10,11,1,39,57,33,51], we employ the COCO dataset [35] to pretrain our model. From Table 6 we can see that our UViT is highly compact in terms of the number of parameters, and achieve competitive mIoU with comparable FLOPs. In addition, it is worth noting that since we only leverage a single-scale feature map, we do not adopt advanced segmentation decoders like ASPP [10] but just use plain convolutional layers for predictions, which is to our disadvantage. Again, without adopting any design conventions from CNNs, the results of our simple UViT architectures suggest that, a simple single-scale transformer backbone can fulfill the dense prediction tasks.

Table 6: Segmentation results on Pascal VOC 2012. Our UViT leverages a plain convolutional segmentation head, without any test-time augmentation.

Backbone	Resolution	GFLOPs	Params. (M)	mIoU
WASPnet-CRF [3]	-	-	47.5	80.4
DeepLabv3+ (ResNet-101) [12]	512×512	298	58.6	79.4
UViT-T (ours)	512×512	163	18.1	79.0
UViT-S (ours)	512×512	215	26.7	79.9
UViT-B (ours)	512×512	350	50.7	81.2
UViT-H (ours)	640×640	3846	529.9	88.1

B Model architectures studied in Figure 2

We show details of all architectures studied in Figure 2 (main body) in Table 7 below. As mentioned in Section 3.1 (main body), we study all combinations of the above three techniques (spatial downsampling "SD", multi-scale features "MF", doubled channels "2×"), i.e. eight settings in total, and show the results in Figure 2. Each dot in Figure 2 indicates an individually designed and trained model. To make all comparisons fair, we carefully design all models such that they are all of around 72 million parameters. We control their FLOPs by changing the depths or attention windows allocated to different stages.

C Model architectures in Figure 4 and Figure 5

We show all architectures studied in our compound scaling rule in Figure 4 and Figure 5 (main body). All models are of 2^{-1}-scale attention windows for fair comparisons.

Table 7: Model architectures in Figure 2 (main body), all studied under a 640×640 input size on MS-COCO. "SD": spatial downsampling. "MF": multi-scale features. "2×": doubled channels. Without any of these three techniques (first section in this table), the whole network has a constant feature resolution and hidden size; all other seven settings below will split the network into three stages, since they require either a progressive feature downsampling or multi-scale features from each stage. Input scale is relative to the 2D shape of the input image $H \times W$ (e.g. 8^{-1} indicates the 2D shape of the UViT's sequence feature is $\frac{1}{8}H \times \frac{1}{8}W$). The window scale is relative to the 2D shape of sequence feature's $h \times w$ (e.g. 8^{-1} indicates the 2D shape of the attention window is $\frac{1}{8}h \times \frac{1}{8}w$). Numbers with underscores in the column "Output Scale" indicate feature maps that will be fed into the FPN detection head (i.e., the last output of backbone if no "MF" is applied, or features from all three stages if "MF" is applied).

SD	MF	2×	Input Scale	#Layers	Window Scale	Hidden Size	Output Scale	Params. (M)	FLOPs (G)	mAP
			8^{-1}	18	16^{-1}	384	$\underline{8^{-1}}$	72.1	534.1	44.5
					8^{-1}				540.9	48.2
					4^{-1}				567.9	50.1
					2^{-1}				676.2	50.7
					1				1109.1	50.8

SD	MF	2×	Stage 1					Stage 2					Stage 3					Params. (M)	FLOPs (G)	mAP
			Input Scale	#Layers	Window Scale	Hidden Size	Output Scale	Input Scale	#Layers	Window Scale	Hidden Size	Output Scale	Input Scale	#Layers	Window Scale	Hidden Size	Output Scale			
✓			8^{-1}	6	1	384	8^{-1}	16^{-1}	6	1	384	16^{-1}	32^{-1}	6	1	384	$\underline{32^{-1}}$	72.1	607.1	41.0
✓				8					5					5					688.28	42.0
✓				10					4					4					769.47	42.6
✓				12					3					3					850.68	43.0
✓				14					2					2					931.88	43.4
	✓		8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1} 1	384	$\underline{8^{-1}}$	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1} 1	384	$\underline{16^{-1}}$	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1} 1	384	$\underline{32^{-1}}$	72.1	534.3	44.3
	✓																		541.03	47.6
	✓																		568.09	49.4
	✓																		676.33	50.3
	✓																		1109.3	50.2
		✓	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1} 1	152	8^{-1}	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1} 1	304	8^{-1}	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1} 1	608	$\underline{8^{-1}}$	73.8	558.4	43.4
		✓																	561.5	44.4
		✓																	587.7	46.3
		✓																	692.2	46.6
		✓																	1110.2	48.3
✓	✓		8^{-1}	2	1	384	$\underline{8^{-1}}$	16^{-1}	8	1	384	$\underline{16^{-1}}$	32^{-1}	8	1	384	$\underline{32^{-1}}$	72.1	459.7	45.8
✓	✓			4					7					7					540.9	47.5
✓	✓			6					6					6					622.1	48.5
✓	✓			8					5					5					703.3	48.0
✓	✓			10					4					4					784.5	48.6
✓	✓			12					3					3					865.7	50.2
✓	✓			15					2					1					989.5	50.4
✓		✓	8^{-1}	16	1	128 160 192 224 256	8^{-1}	16^{-1}	1	1	256 320 384 448 512	16^{-1}	32^{-1}	9 5 3 2 1	1	512 640 768 896 1024	$\underline{32^{-1}}$	70.2 69.3 69.3 71.4 69.2	529.1 581.7 637.4 696.6 756.5	37.6 38.9 40.2 41.7 42.5
✓		✓																		
✓		✓																		
✓		✓																		
✓		✓																		
	✓	✓	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1}	152	$\underline{8^{-1}}$	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1}	304	$\underline{16^{-1}}$	8^{-1}	6	16^{-1} 8^{-1} 4^{-1} 2^{-1}	608	$\underline{32^{-1}}$	73.8	566.3	45.7
	✓	✓																	569.5	46.4
	✓	✓																	595.6	48.1
	✓	✓																	700.1	49.0
✓	✓	✓	8^{-1}	16	1	128 160 192 224 256	$\underline{8^{-1}}$	16^{-1}	1	1	256 320 384 448 512	$\underline{16^{-1}}$	32^{-1}	9 5 3 2 1	1	512 640 768 896 1024	$\underline{32^{-1}}$	73.3 72.4 72.4 74.5 72.4	552.1 604.9 660.7 719.9 779.9	44.3 45.5 47.6 48.8 49.4
✓	✓	✓																		
✓	✓	✓																		
✓	✓	✓																		
✓	✓	✓		28		224					448			1		896		72.1	992.3	49.5

Table 8: Model architectures in Figure 4 and Figure 5 (MS-COCO) (in main body). Configurations (depth, width) of UViT-T/S/B are annotated.

Input Size	Depth	Width	Params. (M)	FLOPs (G)	mAP
640×640	18	384	72.1	676.2	50.4
		432	80.9	748.3	50.5
		462	86.9	796.6	50.7
		492	93.3	847.4	50.4
		564	110.2	979.4	50.1
768×768	18	288	58.2	725.9	51.1
		306	60.7	761.1	51.5
		330	64.3	810.0	51.3
		384	73.1	928.5	51.5
		432	82.1	1043.5	51.6
		462	88.2	1120.1	51.3
896×896	18	186	47.4	710.2	51
		222 (UViT-T)	51.0	801.4	51.3
		246	53.8	866.1	51.7
		288 (UViT-S)	59.2	986.8	51.7
		330	65.4	1117.1	52.1
		384 (UViT-B)	74.4	1298.7	52.3
1024×1024	18	120	42.6	710.3	47.9
		132	43.5	750.1	48.9
		144	44.4	791.0	49.3
		162	45.8	854.3	50.4
		198	49.3	987.6	51.4
		246	54.7	1179.7	51.7
		288	60.3	1361.2	52.0
896×896	12	276	52.1	748.4	50.8
		300	54.4	796.2	50.8
		324	56.9	846.2	51.0
		360	60.9	925.0	51.5
		390	64.5	994.2	51.5
896×896	24	156	46.5	739.0	50.6
		180	49.2	813.8	50.8
		192	50.6	852.7	51.3
		258	60.1	1085.4	51.8
		294	66.3	1225.7	51.6
896×896	32	120	44.6	732.5	50
		132	45.9	777.4	50.4
		144	47.3	823.8	51.2
		180	52.3	971.1	51.5
		240	62.8	1244.4	52.0
896×896	40	96	43.2	723.2	48.5
		102	43.8	749.3	49.1
		114	45.2	802.9	50.1
		126	46.8	858.2	50.7
		150	50.3	974.0	51.2
		156	51.2	1004.0	51.2