

DPT: Deformable Patch-based Transformer for Visual Recognition

Zhiyang Chen^{1,2}, Yousong Zhu¹, Chaoyang Zhao^{1*}, Guosheng Hu³, Wei Zeng⁴, Jinqiao Wang^{1,2}, Ming Tang¹

¹National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China

²School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China

³AnyVision, Belfast, UK

⁴Peking University, Beijing, China

{zhiyang.chen,yousong.zhu,chaoyang.chao,jqwang,tangm}@nlpr.ia.ac.cn

huguosheng100@gmail.com, weizeng@pku.edu.cn

ABSTRACT

Transformer has achieved great success in computer vision, while how to split patches in an image remains a problem. Existing methods usually use a fixed-size patch embedding which might destroy the semantics of objects. To address this problem, we propose a new Deformable Patch (DePatch) module which learns to adaptively split the images into patches with different positions and scales in a data-driven way rather than using predefined fixed patches. In this way, our method can well preserve the semantics in patches. The DePatch module can work as a plug-and-play module, which can easily be incorporated into different transformers to achieve an end-to-end training. We term this DePatch-embedded transformer as Deformable Patch-based Transformer (DPT) and conduct extensive evaluations of DPT on image classification and object detection. Results show DPT can achieve 81.9% top-1 accuracy on ImageNet classification, and 43.7% box mAP with RetinaNet, 44.3% with Mask R-CNN on MSCOCO object detection. Code has been made available at: <https://github.com/CASIA-IVA-Lab/DPT>.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision**; **Image representations**; *Object recognition*; *Object detection*.

KEYWORDS

vision transformer; deformable patch; image classification; object detection

ACM Reference Format:

Zhiyang Chen^{1,2}, Yousong Zhu¹, Chaoyang Zhao^{1*}, Guosheng Hu³, Wei Zeng⁴, Jinqiao Wang^{1,2}, Ming Tang¹. 2021. DPT: Deformable Patch-based Transformer for Visual Recognition. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21)*, October 20–24, 2021, Virtual Event, China. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3474085.3475467>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MM '21, October 20–24, 2021, Virtual Event, China

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-8651-7/21/10...\$15.00

<https://doi.org/10.1145/3474085.3475467>

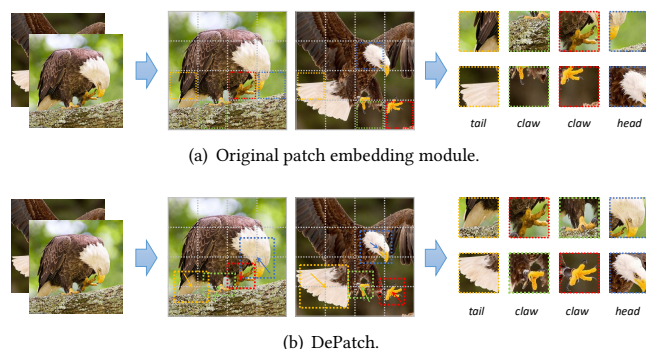


Figure 1: An example of vanilla patch splitting and our deformable way. (a) Original patch embedding module divides the image in a fixed way. It sometimes destroys the semantics of objects. (b) Our DePatch splits the image into patches in a deformable way with learnable positions and scales. (Better viewed in color)

1 INTRODUCTION

Recently, transformer [25] has made significant progress in natural language process (NLP) and speech recognition. It has gradually become the prevailing method for sequence modeling tasks. Inspired by this, some studies have successfully applied transformer in computer vision, and achieved promising performance on image classification [17, 24], object detection [2, 35], and semantic segmentation [33]. Similar to NLP, transformer usually divides the input image into a sequence of fixed-size patches (e.g. 16×16) [7, 24, 26, 30], and models the context relationships between different patches through multi-head self-attention. Compared to convolution neural networks (CNNs), transformer can effectively capture long-range dependency inside the sequence, and the features extracted contain more semantic information.

Although transformer has tasted sweetness in vision tasks, there are still many aspects to be improved. DeiT [24] exploits data augmentation and knowledge distillation to learn visual transformer in

* Corresponding author.

a data-efficient manner. T2T-ViT [30] decomposes the patch embedding module by recursively aggregating neighboring tokens for better local representation. TNT [10] maintains a fine-grained branch to better model local details inside a patch. PVT [26] transforms the architecture into four stages, and generates feature pyramid for dense prediction. These works use a *fixed-size* patch embedding under an implicit assumption that the fixed image split design is suitable for all images. However, such ‘hard’ patch split may bring two problems: (1) Collapse of local structures in an image. As shown in Figure 1(a), a regular patch (16×16) is always hard to capture the complete object-related local structure, since objects are with various scales in different images. (2) Semantic inconsistency across images. The same object in different images might have different geometric variations (scale, rotation, etc). The fixed way of splitting images will potentially capture inconsistent information for one object in different images. As discussed, these fixed patches can potentially destroy the semantic information, leading to degraded performance.

To address the aforementioned problems, in this paper, we propose a new module, called DePatch, which divides images in a deformable way. In this way, we can well preserve the semantics in one patch, reducing the semantics destruction caused by image splitting. To achieve this, we learn the offset and scale of each patch in feature map space. The offset and scale are learned based on the input feature map and are generated for each patch as illustrated in Figure 1(b). The proposed module is lightweight and introduces a very small number of parameters and computations. More importantly, it can work as a plug-and-play module which can easily be incorporated into other transformer architectures. A transformer with DePatch module is named Deformable Patch-based Transformer, DPT. In this work, we integrate DePatch module to Pyramid Vision Transformer (PVT) [26] to verify its efficacy since PVT achieves state-of-the-art performance in pixel-level prediction tasks like object detection and semantic segmentation. With deformable patch adjustment, DPT generates complete, robust and discriminative features for each patch based on local contextual structures. Therefore it can not only achieve high performance on classification tasks, but also outperform other methods on tasks which highly depend on local features, e.g. object detection. Our method achieves 2.3% improvements on ImageNet classification, and improves box mAP by 2.8%/3.5% with RetinaNet and Mask R-CNN framework for MSCOCO object detection compared to its counterpart, PVT-Tiny.

Our main contributions can be summarized as:

- We introduce a new adaptive patch embedding module, DePatch. DePatch can adjust the position and scale of each patch based on the input image, and effectively preserve semantics in one patch, reducing semantics destruction caused by image splitting.
- Our DePatch is lightweight and can work as a plug-and-play module integrated into different transformers, leading to a Deformable Patch-based Transformer (DPT). In this work, we incorporate DePatch into PVT to verify the efficacy of DPT.
- We conduct extensive experiments on image classification and object detection. For example, our module improves

top-1 accuracy by 2.3% on ImageNet classification, and also gains 2.8%/3.5% improvements for both RetinaNet and Mask R-CNN detectors under the tiny configuration.

2 RELATED WORK

2.1 Vision Transformer

Transformer [25] has been the mainstream approach for NLP tasks. It uses self-attention to capture long-range dependence within the whole sequence, and achieves state-of-the-art performance. This idea is applied into computer vision firstly by Non-Local block and its variants [1, 14, 27]. Recently, there appear a large amount of works building pure vision transformers without convolution layers. ViT [7] is as far as we know the first work in this trend. It achieves comparable results with traditional CNN architectures with the help of large training data. DeiT [24] uses complex training schedules and knowledge distillation to improve performance trained on ImageNet only.

Current works focus on combining the advantages of transformer and CNN in order to capture better local information. This object is obtained by combining convolution blocks and self-attention layers together [3, 22], maintaining high-resolution feature maps [10, 17, 26], adding parameters biased for locality [6, 9] or re-designing the brute-force patch-embedding module [30]. Though large improvements achieved, most architectures split the input image with a fixed pattern, without awareness of the input content and geometric variations. Our DPT can modify the position and scale of each patch in an adaptive way. To the best of our knowledge, our model is the first vision transformer that do patch splitting in a data-specific way.

2.2 Deformable-Related Work

Modifying fixed pattern into an adaptive way is a common idea to improve performance. There have been a great number of works help models focus on important features and adopt geometric variations in computer vision. All related works fall into two categories, attention-based methods [1, 13, 14, 27] and offset-based methods [5, 8, 28, 34, 35]. We mainly review offset-based ones.

Offset-based methods predict offsets to explicitly direct important locations. This idea bears some similarity with region proposal network in object detection [11, 20]. Unlike our task, region proposal network uses supervision of bounding box annotations. In image classification task, there are also some works explicitly learning positions of the important regions for better performance [8] or faster inference [28]. The learning process are merely guided by cross-entropy loss and final accuracy. Deformable convolution [5, 34] is the work most similar to ours. It predicts an offset for each pixel of the convolution kernel, while the predicted regions in our method are more regular ones. Irregular patches are not compatible in vision transformers. Deformable-DETR [35] applies deformable operations in the self-attention layers and cross-attention layers of DETR. However, its main purpose is to accelerate training, and Deformable-DETR still relies on feature maps extracted from CNNs. As far as we know, our work is the first to apply deformable operations in a pure vision transformer architecture. We focus on adjusting the position and scale of each patch, therefore extracting features better maintaining local structures. Our module can work

as a plug-and-play module, and is compatible for various vision transformer architectures.

3 METHOD

3.1 Preliminaries: Vision Transformer

Vision transformer is composed of three parts, a patch embedding module, multi-head self-attention blocks and feed-forward multi-layer perceptrons (MLP). The network starts with the patch embedding module which transforms the input image into a sequence of tokens, and then alternately stacks multi-head self-attention blocks and MLPs to obtain the final representation. We mainly elaborate on the patch embedding module in this section, and then have a quick review over the multi-head self-attention.

Patch embedding module divides images into patches with fixed size and positions, and embeds each of them with a linear layer. We denote the input image or feature map as $A \in \mathbb{R}^{H \times W \times C}$. For simplicity, we assume $H = W$. Previous works split A into a sequence of N patches with size $s \times s$ ($s = \lceil H/\sqrt{N} \rceil$). The sequence is denoted as $\{z^{(i)}\}_{1 \leq i \leq N}$.

To better interpret the patch splitting process, we reformulate the patch embedding module. Each patch $z^{(i)}$ can be seen as a representation for a rectangle region of the input image. We denote its center coordinate as $(x_{ct}^{(i)}, y_{ct}^{(i)})$. Since patch size is fixed, the left-top corner and right-bottom corner are determined as $(x_{ct}^{(i)} - s/2, y_{ct}^{(i)} - s/2)$ and $(x_{ct}^{(i)} + s/2, y_{ct}^{(i)} + s/2)$. There are $s \times s$ pixels inside this region, their coordinates are represented by $\tilde{p}^{(i,j)} = (p_x^{(i,j)}, p_y^{(i,j)})$. All coordinates $\tilde{p}^{(i,j)}$ are integers themselves. The features at these pixels are denoted as $\{\tilde{a}^{(i,j)}\}_{1 \leq j \leq N}$. These features are then flattened and processed by a linear layer to get representation for the new patch, as shown in Eq. (1).

$$z^{(i)} = W_{patch} \cdot \text{concat}\{\tilde{a}^{(i,1)}, \dots, \tilde{a}^{(i,s \times s)}\} + b_{patch} \quad (1)$$

Multi-head self-attention module aggregates relative information over the whole input sequence, giving each token a global view. This module learns three groups of representative features for each head, query ($Q_h \in \mathbb{R}^{N \times d}$), key ($K_h \in \mathbb{R}^{N \times d}$) and value ($V_h \in \mathbb{R}^{N \times d}$). Q_h and K_h are multiplied to obtain the attention map $Attn_h$, which represents similarity between different patches. The attention map is used as the weights to sum up V_h . Independent results are calculated for different heads to get more variant features. Results from all heads are then concatenated and transformed to become new representations Z' .

$$Attn_h = \text{Softmax}(Q_h K_h^T / \sqrt{d}) \quad (2)$$

$$Z' = \text{Concat}\{Attn_1 V_1, \dots, Attn_H V_H\} W_{proj} + b_{proj} \quad (3)$$

3.2 DePatch Module

The patch embedding process described in 3.1 is fixed and inflexible. Positions $(x_{ct}^{(i)}, y_{ct}^{(i)})$ and size s are fixed, therefore the rectangle region is unchangeable for each patch. The feature for each patch is directly represented with its inside pixels. In order to better locate important structures and handle geometric deformation, we loosen these constraints to develop our deformable patch embedding module, DePatch.

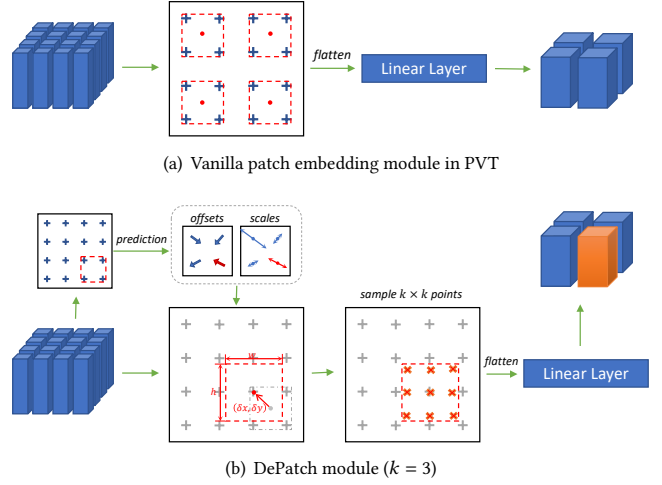


Figure 2: DePatch module instruction. Offsets and scales are predicted with local features, and new embeddings are obtained by bilinear interpolation.

Firstly, we turn the location and the scale of each patch into predicted parameters based on input contents. As for the location, we predict an offset $(\delta x, \delta y)$ allowing it to shift around the original center. As for the scale, we simply replace the fixed patch size s with predictable s_h and s_w . In this way, we can determine a new rectangle region, and denote its left-top corner as (x_1, y_1) and right-bottom corner as (x_2, y_2) . For clarity, we omit the superscript (i) . We emphasize that $(\delta x, \delta y, s_w, s_h)$ can be different even for patches in a single image.

$$\begin{aligned} x_1 &= x_{ct} + \delta x - \frac{s_w}{2}, \quad y_1 = y_{ct} + \delta y - \frac{s_h}{2} \\ x_2 &= x_{ct} + \delta x + \frac{s_w}{2}, \quad y_2 = y_{ct} + \delta y + \frac{s_h}{2} \end{aligned} \quad (4)$$

As shown in Figure 2, we add a new branch to predict these parameters. Based on the input feature map, we predict $(\delta x, \delta y, s_w, s_h)$ densely for all patches first, and then embed them with predicted regions. Offsets and scales are predicted with Eq. (5) and (6). $f_p(\cdot)$ can be any feature extractor, and here we use a single linear layer. After that, W_{offset} and W_{scale} are followed to predict offsets and scales. At the beginning of training, these weights are initialized to zero. b_{scale} is initialized to make sure each patch focuses on exactly the same rectangle region as the original model.

$$\delta x, \delta y = \text{Tanh}(W_{offset} \cdot f_p(A)) \quad (5)$$

$$s_w, s_h = \text{ReLU}(\text{Tanh}(W_{scale} \cdot f_p(A) + b_{scale})) \quad (6)$$

After the rectangle region is determined, we extract the feature for each patch. The main problem is that regions are with different sizes, and the predicted coordinates are usually fractional. We solve this problem in a sampling-and-interpolation manner. Given the rectangle coordinates (x_1, y_1) and (x_2, y_2) , we sample $k \times k$ points uniformly inside the region, k is a super-parameter for our method, which will be ablated in 4.3. Each sampling location is denoted as $\tilde{p}^{(j)} = (p_x^{(j)}, p_y^{(j)})$ for any $1 \leq j \leq k \times k$. The features of all

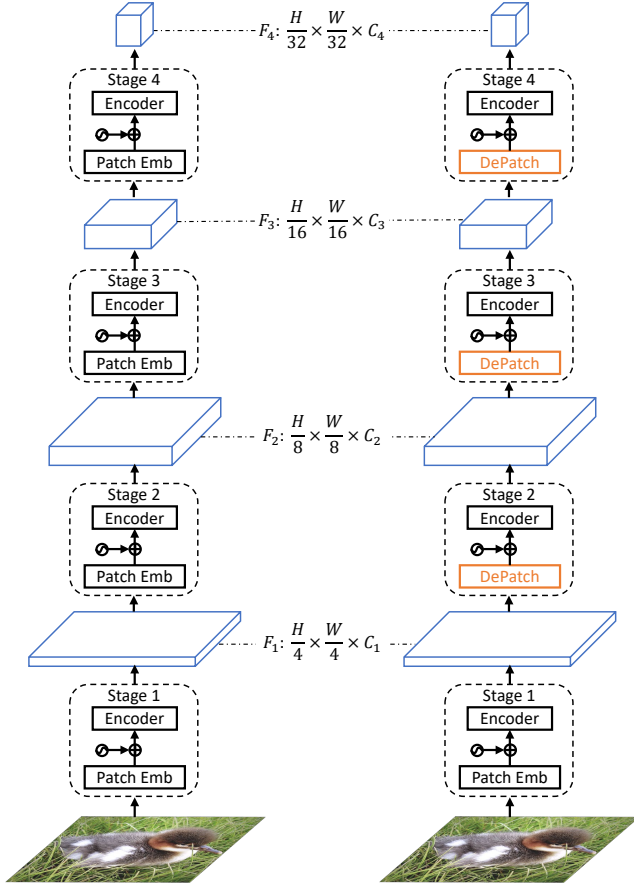


Figure 3: Left: Original PVT architecture. Right: DPT, Equipped with our DePatch module.

sampled points $\{\tilde{a}^{(j)}\}_{1 \leq j \leq k \times k}$ are then flattened and processed in a linear layer to generate patch embedding, as in Eq. (7).

$$z^{(i)} = W \cdot \text{concat}\{\tilde{a}^{(1)}, \dots, \tilde{a}^{(k \times k)}\} + b \quad (7)$$

The index of sampled points is mostly fractional. Assume that we intend to extract feature at point (p_x, p_y) . Its corresponding feature is obtained via bilinear interpolation as

$$A(p_x, p_y) = \sum_{q_x, q_y} G(p_x, p_y; q_x, q_y) \cdot A(q_x, q_y) \quad (8)$$

$$G(p_x, p_y; q_x, q_y) = \max(0, 1 - |p_x - q_x|) \cdot \max(0, 1 - |p_y - q_y|) \quad (9)$$

In Eq. (8), $G(\cdot)$ is the bilinear interpolation kernel all over the integral spatial locations. It only becomes non-zero at the four locations close to (p_x, p_y) . Therefore, it can be computed quickly with few multiply-adds.

3.3 Overall Architecture

DePatch is a self-adaptive module to change positions and scales of patches. As DePatch can work as a plug-and-play module, we can easily incorporate DePatch into various vision transformers. Because of the superiority and generality, we choose PVT as our base model. PVT has four stages with feature maps of decreasing

scales. It utilizes spatial-reduction attention to reduce cost on high resolution feature maps. For detailed information please refer to [26]. Our model is denoted as DPT. It is built by replacing the patch embedding modules at the beginning of stage 2, 3 and 4 with DePatch, while keeping other configurations unchanged. The overall architecture is shown in Figure 3.

4 EXPERIMENTS

In this section, we conduct image classification experiments on ImageNet [21] and object detection experiments on COCO [16]. After that, some ablation studies are provided then for further analysis.

4.1 Image Classification

Experiment Settings We use ImageNet [21] for image classification experiments. The ImageNet dataset consists of 1.28M images for training and 50K for validation. These images belong to 1000 categories. We report top-1 error on validation set for comparison. The images are resized into 256×256 and randomly cropped into 224×224 for training. Advanced data augmentation methods including Mixup [32], CutMix [31], label smoothing [23] and Rand-Augment [4] are utilized. Our models are trained with the batch size of 1024 for 300 epochs and optimized by AdamW [19] with initial learning rate of 1×10^{-3} and cosine schedule [18]. Weight decay is set to 0.05 for non-bias parameters. All these settings keep the same with original PVT [26] for fair comparison.

Results As shown in Table 1, our smallest DPT-Tiny achieves 77.4% top-1 accuracy, which is 2.3% higher than the corresponding baseline PVT model. The best result is achieved by our medium one. It achieves 81.9% top-1 accuracy, and even outperforms models with much larger costs like PVT-Large and catch up with DeiT-Base. As for CNN-based models, DPT-Small outperforms the popular ResNet50 by 4.9%. Our models achieve better results than both CNN-based and transformer-based models, and outperform them by a large margin.

4.2 Object Detection

Experiment Settings Our experiments for object detection are conducted on COCO [16], a large-scale detection benchmark. We set train2017 split with 118K images as our training set, and val2017 split with 5K images for validation. Mean Average Precision (mAP) is used as our evaluation metric. Following [26], we evaluate our DPT backbones on three prevailing frameworks, RetinaNet [15], Mask R-CNN [11] and DETR [2]. We load ImageNet pretrained weights to initialize the backbone. Our models are trained with the batch size of 16 and optimized by AdamW [19] with initial learning rate 1×10^{-4} . As to RetinaNet and Mask R-CNN, we report results with both 1x and multi-scale 3x train schedules (12 and 36 epochs). For 1x schedule, images are resized so that the shorter edge has 800 pixels and the longer edge does not exceed 1333 pixels. For multi-scale training, the shorter edge is resized within the range of [640, 800]. As for DETR, the model is trained for 50 epochs with random flip and random scale.

Results We compare DPT to PVT [26] and standard ResNe(X)t [12, 29]. The comparison is shown in Table 2. As for RetinaNet, DPT-Small significantly outperforms PVT-Small by 2.1% and Resnet50

Table 1: Results on ImageNet Classification.

Method	#Param (M)	FLOPs (G)	Top-1 Acc(%)
ResNet18 [12]	11.7	1.8	69.8
DeiT-Tiny [24]	5.7	1.3	72.2
PVT-Tiny [26]	13.2	1.9	75.1
DPT-Tiny (ours)	15.2	2.1	77.4
ResNet50 [12]	25.6	4.1	76.1
DeiT-Small [24]	22.1	4.6	79.9
T2T-ViT-14 [30]	21.4	5.2	80.6
PVT-Small [26]	24.5	3.8	79.8
DPT-Small (ours)	26.4	4.0	81.0
ResNet101 [12]	44.7	7.9	77.4
X101-32x4d [29]	44.2	8.0	78.8
X101-64x4d [29]	83.5	15.6	79.6
ViT-Base [7]	86.6	17.6	77.9
DeiT-Base [24]	86.6	17.6	81.8
T2T-ViT-19 [30]	39.0	8.0	81.2
PVT-Medium [26]	44.2	6.7	81.2
PVT-Large [26]	61.4	9.8	81.7
DPT-Medium (ours)	46.1	6.9	81.9

by 6.2% mAP at a comparable computational cost, which indicates that DPT provides more discriminative features for target objects in images. With our DePatch module, each patch is aware of its neighboring content, and extracts crucial information needed for different locations. Moreover, with $3\times$ training schedule and multi-scale training, RetinaNet+DPT-Medium achieves 43.7% mAP. It outperforms PVT-Medium and ResNet101 by a large margin, and even achieves better performance than PVT-Large model, but with more than 20% cost reduced.

Results for Mask R-CNN are similar. Our DPT-Small model achieves 43.1% box mAP and 39.9% mask mAP under $1\times$ schedule, outperforming PVT-Small by 2.7% and 2.1%. With $3\times$ training schedule and multi-scale training, Mask R-CNN+DPT-Small achieves the best result with 44.4% box mAP and 41.0% mask mAP.

DETR is a latest framework for object detection. It requires a long trained schedule (e.g. 500 epochs). We only validate our model with a shorter schedule (50 epochs). According to Table 4, our DPT-Small achieves 37.7% box mAP, outperforming PVT-Small by 3.0% and ResNet50 by 5.4%. Therefore we conclude that DPT is also compatible with transformer-based detectors.

4.3 Ablation Studies

Effect of module position There are four patch embedding modules in PVT. The first directly operates on the input image, and the rest are inserted at the beginning of the following stages. We perform detailed experiments to illustrate where we should add DePatch.

Since raw images contain little semantic information, it is difficult for the first module to predict offsets and scales beyond its own region. Therefore, we only attempt to replace the rest three patch embedding modules. The results are shown in Table 5. The improvements obtained by stage 2, 3 and 4 are 0.3%, 1.0%, and 1.5%. The more

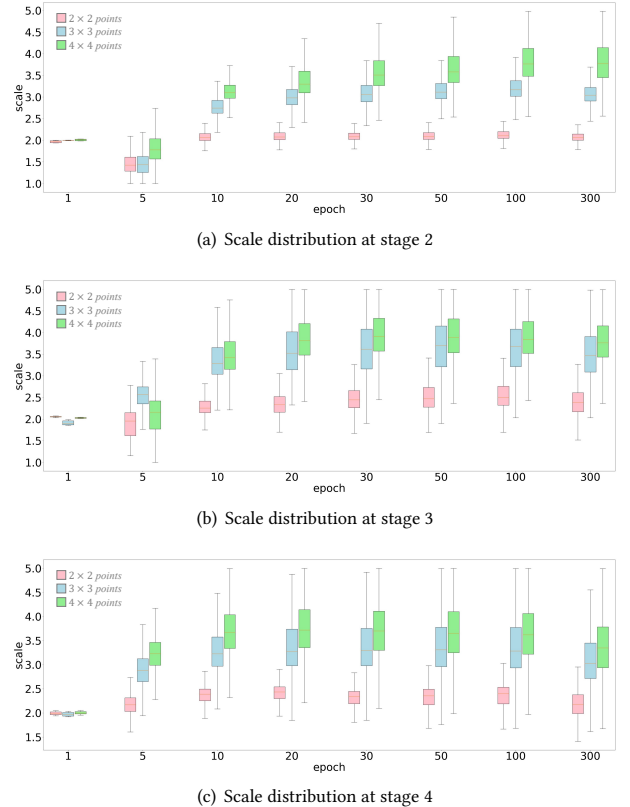


Figure 4: Region scale learned by DPT-Tiny with different number of sampling points $k \times k$. We illustrate statistics in stage 2, 3 and 4. Scale is measured by the edge size of the region.

patch embedding modules we replace, larger improvement it brings. According to the results, replacing all patch embedding modules in stage 2, 3 and 4 achieves the best performance. It outperforms baseline PVT model by 1.5%, with only 0.86M parameters increased. In the following experiments, it will be our default configuration to replace all three patch embedding modules.

Effect of number of sampling points We experiment to show how many points we should sample in one predicted region. Sampling more points slightly increases FLOPs, but also has a stronger learning ability to capture features from a larger region. The results are shown in Table 6. Increasing sampling points from 2×2 to 3×3 provides another 0.8% improvement, while further increasing it to 4×4 only improves by 0.2%. Since sampling 4×4 points only gets marginal improvement. We take the $k = 3$ as the default configuration in following experiments.

We claim that sampling more points will benefits DPT with stronger ability to extract features from larger area. We show how the distributions of predicted scales change during training with different number of sampling points in Figure 4. Although we initialize the region scales strictly the same as that in PVT (*patch_size* = 2), DePatch can learn to expand on its own. This phenomenon accords to the common sense in CNN, that enlarging the receptive field

Table 2: Object detection performance on MS COCO (RetinaNet)

Backbone	#Param(M)	RetinaNet 1×						RetinaNet 3× + MS					
		mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet18 [12]	21.3	31.8	49.6	33.6	16.3	34.3	43.2	35.4	53.9	37.6	19.5	38.2	46.8
PVT-Tiny [26]	23.0	36.7	56.9	38.9	22.6	38.8	50.0	39.4	59.8	42.0	25.5	42.0	52.1
DPT-Tiny (ours)	24.9	39.5	60.4	41.8	23.7	43.2	52.2	41.2	62.0	44.0	25.7	44.6	53.9
ResNet50 [12]	37.7	36.3	55.3	38.6	19.3	40.0	48.8	39.0	58.4	41.8	22.4	42.8	51.6
PVT-Small [26]	34.2	40.4	61.3	43.0	25.0	42.9	55.7	42.2	62.7	45.0	26.2	45.2	57.2
DPT-Small (ours)	36.1	42.5	63.6	45.3	26.2	45.7	56.9	43.3	64.0	46.5	27.8	46.3	58.5
ResNet101 [12]	56.7	38.5	57.8	41.2	21.4	42.6	51.1	40.9	60.1	44.0	23.7	45.0	53.8
ResNeXt101-32x4d [29]	56.4	39.9	59.6	42.7	22.3	44.2	52.5	41.4	61.0	44.3	23.9	45.5	53.7
ResNeXt101-64x4d [29]	95.5	41.0	60.9	44.0	23.9	45.2	54.0	41.8	61.5	44.4	25.2	45.4	54.6
PVT-Medium [26]	53.9	41.9	63.1	44.3	25.0	44.9	57.6	43.2	63.8	46.1	27.3	46.3	58.9
PVT-Large [26]	71.1	42.6	63.7	45.4	25.8	46.0	58.4	43.4	63.6	46.1	26.1	46.0	59.5
DPT-Medium (ours)	55.9	43.3	64.6	45.9	27.2	46.7	58.6	43.7	64.6	46.4	27.2	47.0	58.4

Table 3: Object detection performance on MS COCO (Mask R-CNN)

Backbone	#Param(M)	Mask R-CNN 1×						Mask R-CNN 3× + MS					
		mAP^b	AP_{50}^b	AP_{75}^b	mAP^m	AP_{50}^m	AP_{75}^m	mAP^b	AP_{50}^b	AP_{75}^b	mAP^m	AP_{50}^m	AP_{75}^m
ResNet18 [12]	31.2	34.0	54.0	36.7	31.2	51.0	32.7	36.9	57.1	40.0	33.6	53.9	35.7
PVT-Tiny [26]	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
DPT-Tiny (ours)	34.8	40.2	62.8	43.8	37.7	59.8	40.4	42.2	64.4	46.1	39.4	61.5	42.3
ResNet50 [12]	44.2	38.0	58.6	41.4	34.4	55.1	36.7	41.0	61.7	44.9	37.1	58.4	40.1
PVT-Small [26]	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
DPT-Small (ours)	46.1	43.1	65.7	47.2	39.9	62.9	43.0	44.4	66.5	48.9	41.0	63.6	44.2
ResNet101 [12]	63.2	40.4	61.1	44.2	36.4	57.7	38.8	42.8	63.2	47.1	38.5	60.1	41.3
ResNeXt101-32x4d [29]	62.8	41.9	62.5	45.9	37.5	59.4	40.2	44.0	64.4	48.0	39.2	61.4	41.9
ResNeXt101-64x4d [29]	101.9	42.8	63.8	47.3	38.4	60.6	41.3	44.4	64.9	48.8	39.7	61.9	42.6
PVT-Medium [26]	63.9	42.0	64.4	45.6	39.0	61.6	42.1	44.2	66.0	48.2	40.5	63.1	43.5
PVT-Large [26]	81.0	42.9	65.0	46.6	39.5	61.9	42.5	44.5	66.0	48.3	40.7	63.4	43.7
DPT-Medium (ours)	65.8	43.8	66.2	48.3	40.3	63.1	43.4	44.3	65.6	48.8	40.7	63.1	44.1

Table 4: Object detection performance on MS COCO (DETR with 50 epochs)

Backbone	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet50 [12]	32.3	53.9	32.3	10.7	33.8	53.0
PVT-Small [26]	34.7	55.7	35.4	12.0	36.4	56.7
DPT-Small (ours)	37.7	59.2	38.8	15.0	40.3	58.5

Table 5: Effect of module position ($k = 2$)

Stage 2	Stage 3	Stage 4	#Params(M)	Top-1 Acc(%)
			13.23	75.1
✓			13.26	75.4
✓	✓		13.43	76.1
✓	✓	✓	14.09	76.6

Table 6: Effect of number of sampling points

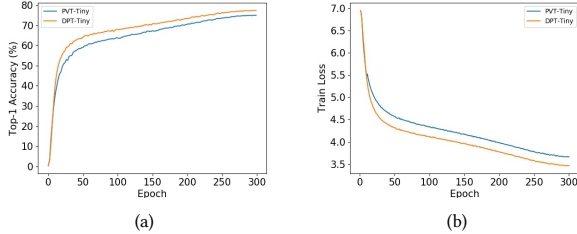
Sampling points	#Params(M)	Flops(G)	Top-1 Acc(%)
Baseline	13.23	1.94	75.1
2×2	14.09	2.03	76.6
3×3	15.15	2.14	77.4
4×4	16.64	2.30	77.6

Table 7: Decouple of predicting offsets and scales

Offsets	Scales	Top-1 Acc(%)
		75.1
✓		76.6
✓	✓	77.4

Table 8: Top-1 accuracy (%) with short training schedule

Method	150 epochs	300 epochs
PVT-Tiny [26]	73.1	75.1
DPT-Tiny (Ours)	76.2	77.4

**Figure 5: Training curve for both DPT-Tiny and PVT-Tiny.**

benefits the model. The distributions of scales for $k = 2$ converge early with low variance. Sampling 2×2 points is unable to represent any larger regions, hence it limits the capacity of the our module for understanding images with heavy geometric deformation. The statistics do not differ much for $k = 3$ and 4 except in stage 2. We assume that sampling more points will achieve even better performance, while it is not worth additional cost. Designing a more sophisticated spatial pyramid could be another way to improve our method. We leave it as our future work.

Decouple offsets and scales DePatch learns both the offset and scale for each patch. Offsets are predicted to shift the patches towards more important regions, and scales are for better maintaining local structures. They both facilitates the performance of our model. We decouple these two factors in Table 7 in order to see how each single one influences our model. When scales are not predicted, the shape of all rectangle regions is fixed the same as patches in original PVT. Only predicting offsets can improve 1.5% above baseline, and another 0.8% is obtained by predicting scales. We claim that both offsets and scales are important for our self-adaptive patch embedding module.

Analysis for fast convergence DePatch module is able to adjust the patches to a proper shape for each image. Adequate patches maintain important local structures, and features are learned more efficiently. Therefore the whole network can learn at a faster speed. We draw the training curve for both our DPT-Tiny and PVT-Tiny in Figure 5. The training loss and test accuracy do converge faster in first few epochs.

Based on this phenomenon, we expect that our module can alleviate the requirement of long training schedule. We prove it by simply reducing training epochs by half. As shown in Table 8, DPT-Tiny trained with only 150 epochs outperforms a fully-trained PVT-Tiny by 1.1%, and the performance degradation caused by a shorter schedule is only 1.2%, which is much smaller than original PVT-Tiny. This indicates that our DePatch module can significantly accelerate training for vision transformers, which would benefit further research.

Table 9: Effect of initialization for W_{offset} and W_{scale}

Initialization	Top-1 Acc(%)
Truncated normal	77.36
Zero init	77.39

Effect of parameter initialization As stated in 3.2, we initialize W_{offset} and W_{scale} to zero. We shown in Table 9 that initialization methods have little impact on final performance. We take zero initialization in the all experiments.

4.4 Visualization

We illustrate offsets learned by DePatch in Figure 6. The visualization shows that patches predicted by DePatch are well-located to capture important features. DePatch has more obvious impacts at the edge of foreground objects. It encourages the patches outside to shift a bit towards the object, thus covering more critical areas than normal patches. When there are more than one object appearing in the image, patches would adjust their positions to the closest one (the two whales in Figure 6(b)). This attribute would be more crucial for object detection, since different patches can be more representative for different objects. Therefore, the detector can better locate and classify all the objects with more related features. The predicted scales are also influenced by richness of local context, such as edges or corners. It becomes small when it needs to focus on subtle details (the beak of the bird in Figure 6(d)), and large if more context is needed (homogenous area of the dog’s stomach Figure 6(h)). The high variance of offsets and scales indicates strong self-adaptability of our method.

5 CONCLUSION

In this paper, we introduce DePatch, a deformable module to split patches. It encourages the network to extract patch information from object-related regions and make our model insensitive to geometric deformation. This module can work as a plug-and-play module and improve various vision transformers. We also build a transformer with DePatch module, named Deformable Patch-based Transformer, DPT. Extensive experiments on image classification and object detection indicate that DPT can extract better features and outperform CNN-based models and other vision transformers. DePatch can be utilized in other vision transformers as well as other downstream tasks to improve their performance. Our model is the first work to modify vision transformer in a data-dependant way. We hope our idea could serve as a good starting point for future studies.

ACKNOWLEDGMENTS

This work was supported by the Research and Development Projects in the Key Areas of Guangdong Province (No.2020B010165001) and National Natural Science Foundation of China under Grants No.61772527, No.61976210, No.62002357, No.61876086, No.61806200, No.62002356, and No.61633002.

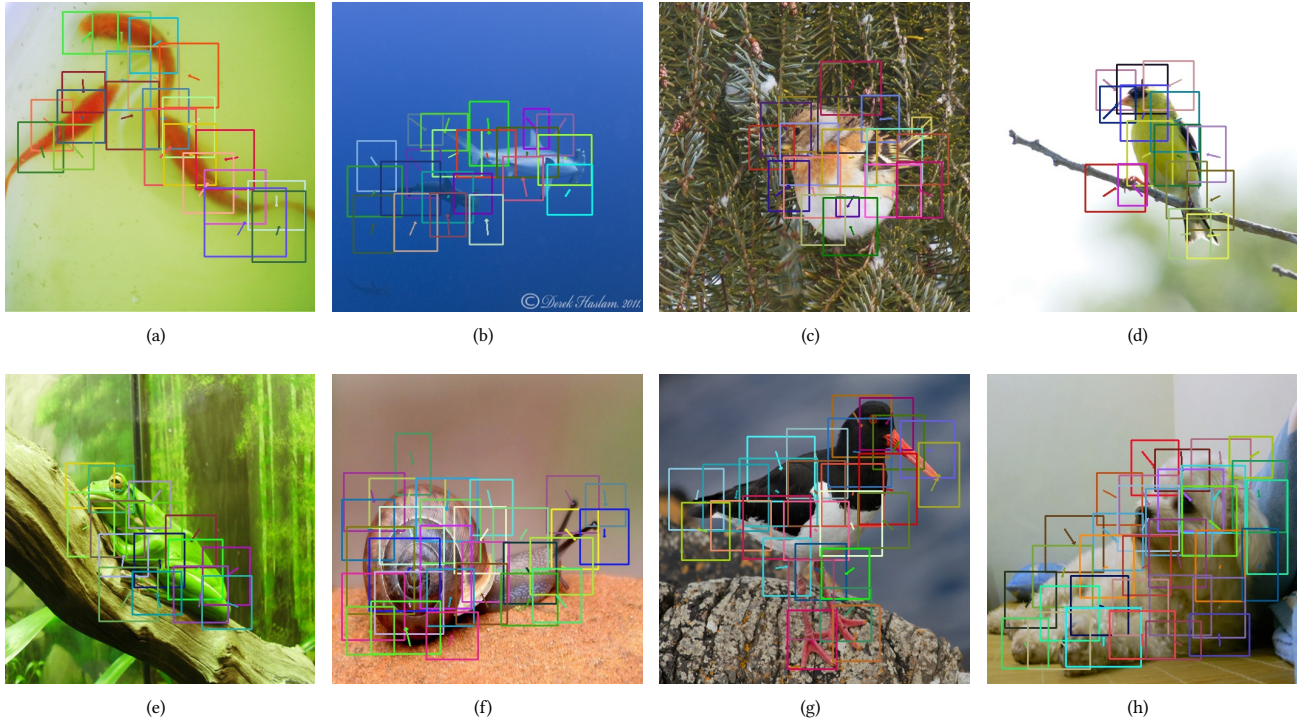


Figure 6: Visualization of learned patches and their offsets with our DPT-Small at stage 4. Our method can adaptively adjust the position and scale for each patch based on the input content.

REFERENCES

- [1] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. 2019. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 0–0.
- [2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European Conference on Computer Vision*. Springer, 213–229.
- [3] Xiangxiang Chu, Bo Zhang, Zhi Tian, Xiaolin Wei, and Huaxia Xia. 2021. Do We Really Need Explicit Position Encodings for Vision Transformers? *arXiv preprint arXiv:2102.10882* (2021).
- [4] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. 2020. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 702–703.
- [5] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. 2017. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*. 764–773.
- [6] Stéphane d’Ascoli, Hugo Touvron, Matthew Leavitt, Ari Morcos, Giulio Biroli, and Levent Sagun. 2021. ConViT: Improving Vision Transformers with Soft Convolutional Inductive Biases. *arXiv preprint arXiv:2103.10697* (2021).
- [7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiahua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
- [8] Jianlong Fu, Heliang Zheng, and Tao Mei. 2017. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4438–4446.
- [9] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. 2021. Fast convergence of detr with spatially modulated co-attention. *arXiv preprint arXiv:2101.07448* (2021).
- [10] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. 2021. Transformer in transformer. *arXiv preprint arXiv:2103.00112* (2021).
- [11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 2961–2969.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
- [13] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.
- [14] Zilong Huang, Xingang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. 2019. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 603–612.
- [15] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.
- [16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.
- [17] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *arXiv preprint arXiv:2103.14030* (2021).
- [18] Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (2016).
- [19] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).
- [20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497* (2015).
- [21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115, 3 (2015), 211–252.
- [22] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. 2021. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605* (2021).
- [23] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.
- [24] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2020. Training data-efficient image transformers

- & distillation through attention. *arXiv preprint arXiv:2012.12877* (2020).
- [25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).
 - [26] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2021. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122* (2021).
 - [27] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. 2018. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7794–7803.
 - [28] Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, Le Yang, and Gao Huang. 2020. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. *arXiv preprint arXiv:2010.05300* (2020).
 - [29] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1492–1500.
 - [30] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. 2021. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986* (2021).
 - [31] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6023–6032.
 - [32] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).
 - [33] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. 2020. Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers. *arXiv preprint arXiv:2012.15840* (2020).
 - [34] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. 2019. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9308–9316.
 - [35] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2020. Deformable DETR: Deformable Transformers for End-to-End Object Detection. *arXiv preprint arXiv:2010.04159* (2020).