Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions — *Supplemental Materials*

Wenhai Wang[1], Enze Xie[2], Xiang Li[3], Deng-Ping Fan[4]✉,
Kaitao Song[3], Ding Liang[5], Tong Lu[1]✉, Ping Luo[2], Ling Shao[4]
[1]Nanjing University [2]The University of Hong Kong
[3]Nanjing University of Science and Technology [4]IIAI [5]SenseTime Research
https://github.com/whai362/PVT

	Output Size	Layer Name	PVT-Tiny	PVT-Small	PVT-Medium	PVT-Large
Stage 1	$\frac{H}{4} \times \frac{W}{4}$	Patch Embedding	$P_1 = 4;\ C_1 = 64$			
		Transformer Encoder	$\begin{bmatrix} R_1 = 8 \\ N_1 = 1 \\ E_1 = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} R_1 = 8 \\ N_1 = 1 \\ E_1 = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} R_1 = 8 \\ N_1 = 1 \\ E_1 = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} R_1 = 8 \\ N_1 = 1 \\ E_1 = 8 \end{bmatrix} \times 3$
Stage 2	$\frac{H}{8} \times \frac{W}{8}$	Patch Embedding	$P_2 = 2;\ C_2 = 128$			
		Transformer Encoder	$\begin{bmatrix} R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix} \times 8$
Stage 3	$\frac{H}{16} \times \frac{W}{16}$	Patch Embedding	$P_3 = 2;\ C_3 = 320$			
		Transformer Encoder	$\begin{bmatrix} R_3 = 2 \\ N_3 = 5 \\ E_3 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} R_3 = 2 \\ N_3 = 5 \\ E_3 = 4 \end{bmatrix} \times 6$	$\begin{bmatrix} R_3 = 2 \\ N_3 = 5 \\ E_3 = 4 \end{bmatrix} \times 18$	$\begin{bmatrix} R_3 = 2 \\ N_3 = 5 \\ E_3 = 4 \end{bmatrix} \times 27$
Stage 4	$\frac{H}{32} \times \frac{W}{32}$	Patch Embedding	$P_4 = 2;\ C_4 = 512$			
		Transformer Encoder	$\begin{bmatrix} R_4 = 1 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} R_4 = 1 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} R_4 = 1 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} R_4 = 1 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix} \times 3$

Table A1: **Detailed settings of Pyramid Vision Transformer (PVT) series.** The design follows the two rules of ResNet [5]. (1) With the growth of network depth, the hidden dimension gradually increases, and the output resolution progressively shrinks; (2) The major computation resource is concentrated in Stage 3.

A1. Details of PVT Series

As described in Sec. 3, the hyper-parameters of our Pyramid Vision Transformer (PVT) are listed as follows:

- P_i: the patch size of Stage i;
- C_i: the channel number of the output of Stage i;
- L_i: the number of encoder layers in Stage i;
- R_i: the reduction ratio of the SRA in Stage i;
- N_i: the head number of the SRA in Stage i;
- E_i: the expansion ratio of the feed-forward layer [9] in Stage i;

Following the design rules of ResNet [5], we (1) use small output channel numbers in shallow stages; and (2) concentrate the major computation resource in intermediate stages. To provide instances for discussion, we describe a series of PVT models with different scales, namely PVT-Tiny, -Small, -Medium, and -Large, in Table A1, whose parameter numbers are comparable to ResNet18, 50, 101, and 152 respectively.

A2. Pure Transformer Semantic Segmentation

We build a pure Transformer model for semantic segmentation by combining our PVT with Trans2Seg [12], a Transformer-based segmentation head. According to the experimental settings in Sec. 5.3, we perform experiments on ADE20K [15] with 40k iterations training, single scale testing, and compare it with ResNet50+Trans2Seg [12] and DeeplabV3+ [2] with ResNet50-d8 (dilation 8) and -d16(dilation 8), as shown in Table A2.

We find that our PVT-Small+Trans2Seg achieves 42.6 mIoU, outperforming ResNet50-d8+DeeplabV3+ (41.5). Note that, ResNet50-d8+DeeplabV3+ has 120.5 GFLOPs

✉ Corresponding authors: Deng-Ping Fan (dengpfan@gmail.com); Tong Lu (lutong@nju.edu.cn).

Method	#Param (M)	GFLOPs	mIoU (%)
ResNet50-d8+DeeplabV3+ [2]	26.8	120.5	41.5
ResNet50-d16+DeeplabV3+ [2]	26.8	45.5	40.6
ResNet50-d16+Trans2Seg [12]	56.1	79.3	39.7
PVT-Small+Trans2Seg	32.1	31.6	42.6(+2.9)

Table A2: **Performance of the pure Transformer semantic segmentation pipeline.** We build a pure Transformer detector by combining PVT and Trans2Seg [12]. It is 2.9% higher than ResNet50-d16+Trans2Seg and 1.1% higher than ResNet50-d8+DeeplabV3+ with lower GFlops. "d8" and "d16" means dilation 8 and 16, respectively.

Method	Scale	GFLOPs	Time (ms)	RetinaNet 1x AP	AP_{50}	AP_{75}
ResNet50 [5]	800	239.3	55.9	36.3	55.3	38.6
PVT-Small (ours)	640	157.2	51.7	38.7	59.3	40.8
	800	285.8	76.9	40.4	61.3	43.0

Table A3: **Latency and AP under different input scales.** "Scale" and "Time" denote the input scale and time cost per image. When the shorter side is 640 pixels, the PVT-Small+RetinaNet has a lower GFLOPs and time cost (on a V100 GPU) than ResNet50+RetinaNet, while obtaining 2.4 points better AP (38.7 *vs*. 36.3).

due to the high computation cost of dilated convolution, and our method has only 31.6 GFLOPs, which is 4 times fewer. In addition, our PVT-Small+Trans2Seg performs better than ResNet50-d16+Trans2Seg (mIoU: 42.6 *vs*. 39.7, GFlops: 31.6 *vs*. 79.3). These results prove that *a pure Transformer segmentation network is workable.*

A3. Supplementary Ablation Study

A3.1. Settings

The experimental settings are the same as those in Sec. 5.5.

A3.2. Speed Analysis

On COCO, the shorter side of the input image is 800 pixels. Under this condition, the inference speed of RetinaNet based on PVT-Small is slower than the ResNet50-based model. A direct solution for this problem is to reduce the input scale. As reported in Table A3, when reducing the shorter side of the input image to 640 pixels, the model based on PVT-Small runs faster than the ResNet50-based model (51.7ms *vs*., 55.9ms), with 2.4 higher AP (38.7 *vs*. 36.3).

A3.3. Deeper *vs*. Wider

The problem of whether the CNN backbone should go deeper or wider has been extensively discussed in previous works [5, 13]. Here, we explore this problem in our Transformer backbone. For fair comparisons, we use PVT-Small as the baseline and design a deeper model and a wider

Method	#Param (M)	Top-1	RetinaNet 1x AP	AP_{50}	AP_{75}
Wider PVT-Small	46.8	19.3	40.8	61.8	43.3
Deeper PVT-Small	44.2	18.8	41.9	63.1	44.3

Table A4: **Deeper *vs*. Wider.** "Top-1" denotes the top-1 error on the ImageNet validation set. "AP" denotes the bounding box AP on COCO `val2017`. The deeper model obtains better performance than the wider model under comparable parameter number.

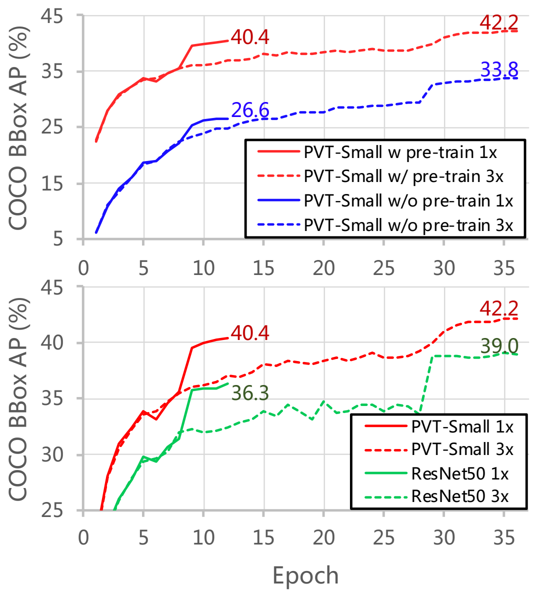

Figure A1: **AP curves of RetinaNet on COCO `val2017` under different backbone settings.** Top: using weights pre-trained on ImageNet *vs*. random initialization. Bottom: PVT-Small *vs*. ResNet50 [5].

model, whose parameters are comparable. For the deeper model, we choose PVT-Medium. For the wider model, we multiply the hidden dimensions $\{C_1, C_2, C_3, C_4\}$ of PVT-Small by a scale factor 1.4, to make it have an equivalent parameter number to the deeper model (*i.e.*, PVT-Medium). As shown in Table A4, the deeper model (*i.e.*, PVT-Medium) consistently works better than the wide model on both ImageNet and COCO. Therefore, going deeper is more effective than going wider in the design of PVT. Based on this observation, in Table A1, we develop PVT models with different scales by simply increasing the model depth.

A3.4. Pre-trained Weights

Most dense prediction models (*e.g.*, RetinaNet [7]) rely on the CNN backbone whose weights are pre-trained on ImageNet. We also discuss this problem in our Transformer bacbkone. At the top of Figure A1, we plot the validation AP curves of PVT-Small+RetinaNet w/ (red curves)

Method	#Param (M)	GFLOPs	Mask R-CNN 1x		
			AP^m	AP^m_{50}	AP^m_{75}
ResNet50+GC r4 [1]	54.2	279.6	36.2	58.7	38.3
PVT-Small (ours)	44.1	304.4	37.8	60.1	40.3

Table A5: **PVT *vs.* CNN w/ non-local.** AP^m denotes mask AP. Under similar parameter nubmer and GFLOPs, our PVT outperform the CNN backbone w/ Non-Local (ResNet50+GC r4) by 1.6 AP^m (37.8 *vs.* 36.2).

and w/o (blue curves) pre-trained weights. We find that the model w/ pre-trained weights converges better than the one w/o pre-trained weights, and the gap between their final AP reaches 13.8 under the 1× training schedule and 8.4 under the 3× training schedule and multi-scale training. Therefore, like CNN-based models, pre-training weights can also help PVT-based models converge faster and better. Moreover, at the bottom of Figure A1, we also see that the convergence speed of PVT-based models (red curves) is faster than that of ResNet-based models (green curves).

A3.5. PVT *vs.* "CNN w/ Non-Local"

To obtain a global receptive field, some well-engineered CNN backbones, such as GCNet [1], integrate the non-local block in the CNN framework. Here, we compare the performance of our PVT (pure Transformer) and GCNet (CNN w/ non-local), using Mask R-CNN for instance segmentation. As reported in Table A5, we find that our PVT-Small outperforms ResNet50+GC r4 [1] by 1.6 points in AP^m (37.8 *vs*. 36.2), and 2.0 points in AP^m_{75} (38.3 *vs*. 40.3), under comparable parameter number and GFLOPs. There are two possible reasons for this result:

(1) Although a single global attention layer (*e.g.*, non-local [11] or multi-head attention (MHA) [9]) can acquire global-receptive-field features, the model performance keeps improving as the model deepens. This indicates that *stacking multiple MHAs can further enhance the representation capabilities of features.* Therefore, as a pure Transformer backbone with more global attention layers, our PVT tends to perform better than the CNN backbone equipped with non-local blocks (*e.g.*, GCNet).

(2) Regular convolutions can be deemed as special instantiations of spatial attention mechanisms [16]. In other words, the format of MHA is more flexible than the regular convolution. For example, for different inputs, the weights of the convolution are fixed, but the attention weights of MHA change dynamically with the input. Thus, *the features learned by the pure Transformer backbone full of MHA layers, could be more flexible and expressive.*

A4. Detection & Segmentation Results

In Figure A2, we also present some qualitative object detection and instance segmentation results on COCO `val2017` [8], and semantic segmentation results on ADE20K [15]. These results indicate that a pure Transformer backbone (*i.e.*, PVT) without convolutions can also be easily plugged in dense prediction models (*e.g.*, RetinaNet [7], Mask R-CNN [4], and Semantic FPN [6]), and obtain high-quality results.

A5. Discussion

Question 1: Compared to the 3×3 convolution, the attention layer has fewer parameters but larger FLOPs. Accordingly, the computational cost of PVT seems to be larger than ResNet [5], when the parameter number is equal.

Answer: The computational cost (FLOPs) depends on the input resolution. As shown in Figure 5, when the input scale is less than 640×640 pixels, the FLOPs of PVT-Small is similar to ResNet50, and significantly lower than ViT-Small [3]. Even when the input scale is up to 800×800 pixels, the FLOPs gap between PVT-Small and ResNet50 is still insignificant. These results indicate that *under the medium input scale, the computational overhead of our model is similar to ResNet50 with a comparable parameter number, and lower than ViT.* There are two reasons as follows:

(1) Unlike the original MHA that requires a large computational resource, *the SRA in our model even has lower FLOPs than the 3×3 convolution with a similar parameter number, when the input scale is small.*

(2) *The main parameters and calculations in our model are contributed by feed-forward layers [9]*, whose parameter number and FLOPs are similar to the common convolution.

Finally, we point out that because the operations of Transformer and CNN are different, it is difficult to keep all indicators (*e.g.*, parameter number and FLOPs) the same. In this work, we mainly discuss the models from the aspect of the number of learnable parameters, which is one of the most important indicators of a model.

Question 2: Why not employ other efficient attention layers (*e.g.*, linear attention [10], sparse attention [14] and deformable attention [17]), and improve the position embedding in PVT?

Answer: In this work, we target to introduce the pyramid framework into Transformer, for dense prediction tasks. To verify the effectiveness of the pyramid structure Transformer, we keep the original Transformer [9] and ViT [3] settings as much as possible when designing our model.

In summary, this work is mainly focused on the overall framework of Transformer, and for the specific modules (*e.g.*, attention layer and position embedding), we would like to leave it to our future work.

Question 3: What are the future directions worth exploring for the vision Transformer backbone?

Answer: There are many potential directions, including but

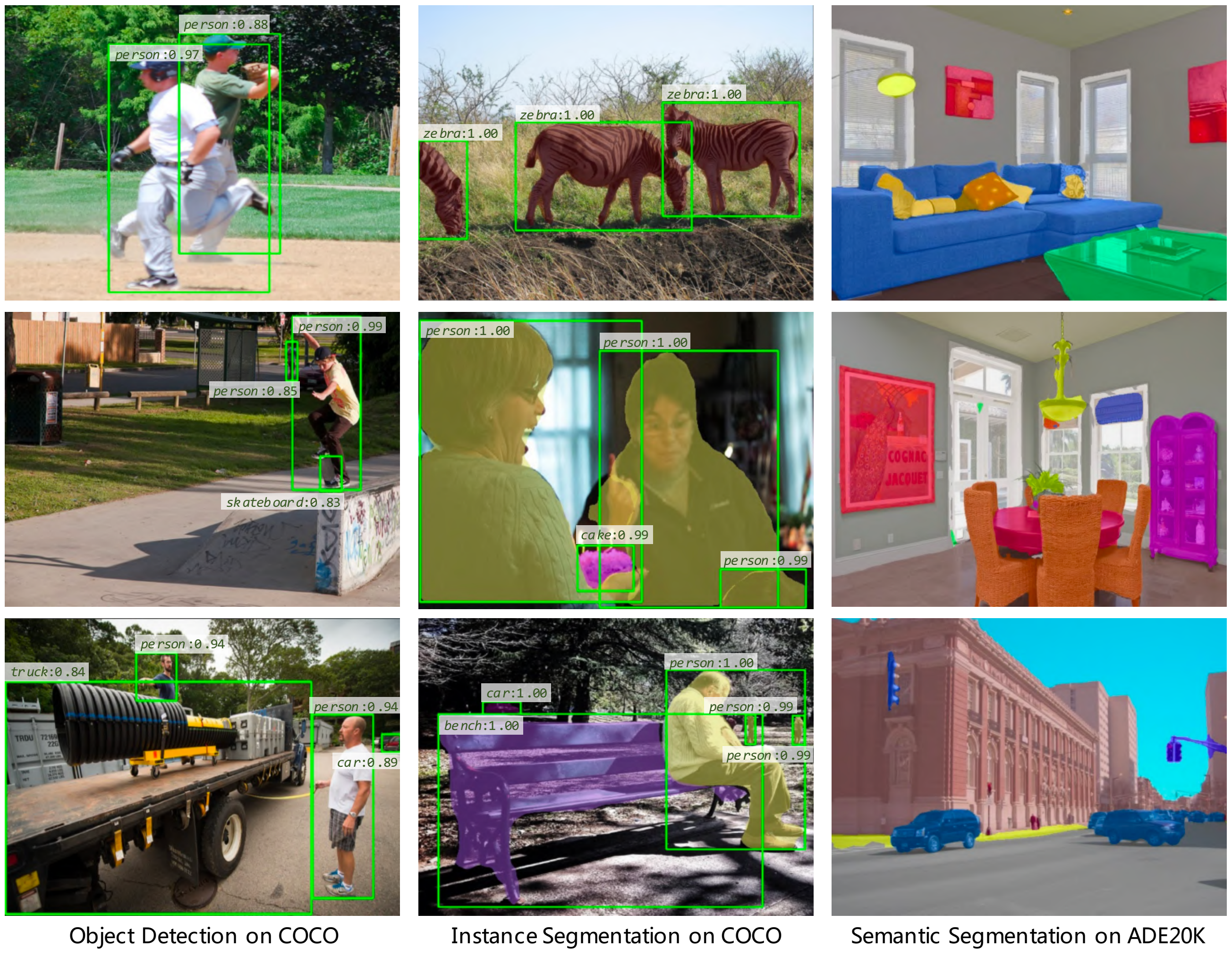
Object Detection on COCO
Instance Segmentation on COCO
Semantic Segmentation on ADE20K

Figure A2: **Qualitative results of object detection and instance segmentation on COCO `val2017` [8], and semantic segmentation on ADE20K [15].** The results (from left to right) are generated by PVT-Small-based RetinaNet [7], Mask R-CNN [4], and Semantic FPN [6], respectively.

not limited to the following three aspects:

(1) **Efficient Attention.** Due to the limitation of the attention layer, it is difficult for the Transformer backbone to process high-resolution input images. Therefore, it is important to explore more effective attention mechanisms for image processing.

(2) **Position Embedding.** Both ViT and PVT use a set of randomly initialized parameters as the position embedding, which is inflexible and sub-optimal for input images of arbitrary resolution. Therefore, a more flexible position embedding for 2D/3D images is required.

(3) **Pyramid Structure.** Our PVT is just a starting point of the pyramid structure Transformer. We believe there are many potential technologies to be explored in the future, and there would be a more elegant and more effective solution than PVT.

References

[1] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2019. 3

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. Eur. Conf. Comp. Vis.*, 2018. 1, 2

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Trans-

formers for image recognition at scale. *Proc. Int. Conf. Learn. Representations*, 2021. 3

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2017. 3, 4

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016. 1, 2, 3

[6] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019. 3, 4

[7] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2017. 2, 3, 4

[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proc. Eur. Conf. Comp. Vis.*, 2014. 3, 4

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. Advances in Neural Inf. Process. Syst.*, 2017. 1, 3

[10] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 3

[11] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. 3

[12] Enze Xie, Wenjia Wang, Wenhai Wang, Peize Sun, Hang Xu, Ding Liang, and Ping Luo. Segmenting transparent object in the wild with transformer. In *Proc. Int. Joint Conf. Artificial Intell.*, 2021. 1, 2

[13] Erwan Zerhouni, Dávid Lányi, Matheus Viana, and Maria Gabrani. Wide residual networks for mitosis detection. In *IEEE International Symposium on Biomedical Imaging*, 2017. 2

[14] Guangxiang Zhao, Junyang Lin, Zhiyuan Zhang, Xuancheng Ren, Qi Su, and Xu Sun. Explicit sparse transformer: Concentrated attention through explicit selection. *arXiv preprint arXiv:1912.11637*, 2019. 3

[15] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017. 1, 3, 4

[16] Xizhou Zhu, Dazhi Cheng, Zheng Zhang, Stephen Lin, and Jifeng Dai. An empirical study of spatial attention mechanisms in deep networks. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2019. 3

[17] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. In *Proc. Int. Conf. Learn. Representations*, 2021. 3