International Journal of Machine Learning and Cybernetics (2023) 14:2127–2136
https://doi.org/10.1007/s13042-022-01750-0

ORIGINAL ARTICLE

Conv-PVT: a fusion architecture of convolution and pyramid vision transformer

Xin Zhang[1] · **Yi Zhang**[1]

Received: 17 June 2022 / Accepted: 10 December 2022 / Published online: 22 December 2022

Abstract
Vision Transformer (ViT) has fully exhibited the potential of Transformer in computer vision domain. However, the computational complexity is proportional to the input dimension which is a constant value for Transformer. Therefore, training a vision transformer network is extremely memory expensive, where a large number of intermediate activation functions and parameters are involved to compute the gradients during back-propagation. In this paper, we propose Conv-PVT (Convolution blocks + Pyramid Vision Transformer) to improve the overall performance of vision transformer. Especially, we deploy simple convolution blocks in the first layer to reduce the memory footprint by down-sampling the input. Extensive experiments (including image classification, object detection and segmentation) have been carried out on ImageNet-1k, COCO and ADE20k datasets to test the accuracy, training time, memory occupation and robustness of our model. The results demonstrate that Conv-PVT achieves comparable performances with the original PVT and outperforms ResNet and ResNetXt for some downstream vision tasks. But it shortens 60% of the training time and reduces 42% GPU (Graphics Processing Unit) memory occupation, realizing twice the inference speed of PVT.

Keywords Attention · Vision transformer · Convolution · Down-stream vision tasks

1 Introduction

Till now, the convolutional neural networks (CNNs) is still the predominant scheme in the field of computer vision. Recently, the rising trend of transformer in natural language processing (NLP) has proved its excellent context modelling abilities, which has aroused great interest of scholars in migrating it to computer vision domain. However, challenges arise in adapting transformer from language to vision when dealing with high resolution of pixels and large variations in scales. Under such circumstances, many works have attempted to replace CNNs with transformer blocks [43, 50] or endeavored to combine CNN-like architectures with attention mechanism to solve fundamental computer vision tasks (e.g. detection, classification, segmentation and tracking etc.). In the meantime, researchers are dedicated to improving the performance and adaptability of transformer with self-attention modules. Among them, ViT [11] was the first successful application of transformer model in image classification. It firstly splits an image into non-overlapping patches (16×16 or 14×14), and provided a sequence of linear embeddings for these patches as input to the transformer. Then those patches were regarded as image representations, which were processed in a similar way as tokens in NLP. Finally, a multi-head self-attention module was utilized to transform the representations into prediction results through a small-scale multi-layer perceptron (MLP). Albeit successful, pure transformer architectures (like ViT) often require a larger amount of training data (or extra supervision) to attain similar performance as CNNs. And they only perform well on large scale dataset than small scale counterpart. DETR [40] reasoned the relation between the objects and global context to yield a set of prediction results given just a small set of learned object queries. The computational cost of the original attention module in transformer architecture is expressed as $O(N^2d)$, which is relatively huge compared with CNNs. Later, many revised versions were published to improve the prediction accuracy of the traditional transformer. However, the improved performance comes at a cost of enormous number of parameters. In view of this, Zhang et al. [52] and Wang et al. [43] optimized the computation

✉ Yi Zhang
yi.zhang@scu.edu.cn

[1] Department of Computer Science, Sichuan University, Chengdu, China

methodology to reduce computation load. But their networks were still huge, which need to be trained on high-performance platforms. For instance, training a PVT-T [43] with a batch size of 128 on ImageNet requires at least 9G GPU memory, and spends nearly 1 hour to train 1 epoch, while PVT-L cannot fit into a 24GB RTX-3090 GPU with the same settings. Furthermore, Swin-T [30] costs 12 mins to train an epoch on 16 V100 GPUs. Except for a few commercial companies, most institutions cannot afford such advanced machines to follow up on this area. It was reported by [34], ViT lacks image-specific inductive biases that is inherent in CNNs (including translation equivariance and locality), it thus does not generalize well when trained on insufficient amount of data. Yuan et al. [49] claimed that image segmentation process causes the loss of edge information among blocks, which results in performance degradation. Xiao et al. [45] merged convolution with vision transformer, and discovered that early convolution boosts the performance of vision transformer.

Inspired by the above works, we develop Conv-PVT in this paper to improve the general performance of the original Transformer in implementing vision related tasks. We aim to reduce the memory footprint and training time while increase the inference accuracy and robustness of the network. To realize the goal, we integrate convolution into the original transformer structure, which down-samples the input image into 1/8 of its original size, and feeds it to transformer encoder. Our model captures the fine-grained features and produces feature maps with rich semantic meanings and content descriptions, which is capable of many dense prediction vision related tasks. We test the accuracy of Conv-PVT on ImageNet-1k. Experiment results validate that the new model only consumes 5G GPU memory and requires 24 mins to train 1 epoch on a RTX 3090 with negligible loss of accuracy.

In a nutshell, the main contributions of our work can be summarized as follows:

(1) We propose Conv-PVT in this paper, which combines the convolution blocks with transformer architecture. Especially, the convolution blocks introduce inductive bias into transformer, which is naturally absent in original transformer architecture. It turns out that our proposed network has far less parameters than existing ones.
(2) We propose to implant Conv-stem to reduce the memory footprint and training time of transformer. It turns out that the Conv-stem not only doubles the validation speed but also achieves higher accuracy through more extensive training in some downstream vision tasks.
(3) We also find that convolution blocks weaken the robustness of transformer, which becomes more notable as the model size grows.

The rest of the paper is organized as follows: related works are discussed in Sect. 2. The architecture of our network is described in Sect. 3 in detail. The experimental results with ablation studies and analysis are shown in Sect. 4. A final conclusion is drawn in Sect. 5 with the corresponding application perspectives (Fig. 1).

2 Related works

Background: Convolution and self-attention are both powerful approaches for representation learning, which follow different design paradigms. CNNs have been widely applied in vision related tasks and achieve state-of-the-art performances. It reaps the benefit of aggregation function over receptive field based on shared weights. The intrinsic properties impose critical inductive bias for image and vision operations. By contrast, self-attention was firstly developed to accomplish natural language processing tasks, which later ignited the great interest of computer vision specialists. It employs weighted average operations to process context of input images so as to capture informative features.

Convolutional neural networks (CNNs): CNNs capture and well-fit the fine-grained features with the properties of weight sharing, rotational and translational invariance. Since the great success of AlexNet [25] in classification on ImageNet, CNNs are the de-facto standard in vision related fields. With the recent development of high-performance GPU, significant improvement has been made by stacking deeper convolutional layers [17, 36]. In addition, many lightweight models [8, 21, 53] were also announced with much less computations to cater for the ever-growing needs of mobile applications. At the same time, different training

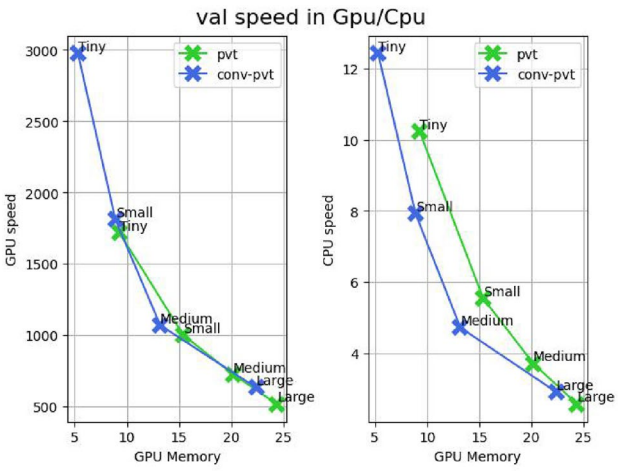

Fig. 1 The X-axis is the GPU Memory usage during training, the Y-axis is the validation speed in CPU or GPU

paradigms [4, 15] and data processing schemes [38, 39] have been put forward to further enhance the behavior of CNNs.

Transformers: Unlike CNNs, transformer builds global relation of context based on self-attention mechanism. It was firstly developed by [41] for NLP topics. BERT elaborated self-supervised pre-training method, while GPT [33] outlined discriminative fine-tuning scheme for specific task with minimal changes to the model architecture. A typical transformer consists of an encoder and a decoder, both of which utilize multi-head attention and feature pyramid network (FPN). The decoder receives the output of the encoder and the original input to predict next sequence. Transformer learns long-distance dependencies quite well and is thus tailored for NLP problems.

Vision Transformers: Enlightened by the success of transformer in NLP, scientists attempted to migrate it to image and vision field. But the early applications were hindered by its huge parameters of network and the quadratic computational complexity. With the advent of high speed hardware, Dosovitskiy et al. [11]presented ViT, where the transformer encoder partitioned input images into non-overlapping patches and projected them into specific dimension to generate the token. Then, they utilized multi-head self-attention module and feed-forward module to model these tokens. The main disadvantage of ViT [11] is its heavy-weight, which only performs well on large scale benchmarks (e.g. JFT-300M, ImageNet-22k), but less satisfactory on small scale dataset (e.g. ImageNet-1k). Additionally, since it is utilized by fixed length coding scheme, it cannot process high-resolution input. Many works tried to train transformer on small scale benchmarks (e.g. CIFAR or ImageNet). DEiT [40] leveraged the merits of distillation and explored data-efficient training scheme. T2T-ViT [50] created token using overlapped sliding window. Swin-Transformer [30] advocated a shifted windowing method, which improved the efficiency by restricting self-attention computation to non-overlapping local windows while realizing cross-window connection at the same time. Zhang et al. [54] aggregated spatio-temporal information through stacked attentions. Guo et al. [16] enhanced input embedding via farthest point sampling and nearest neighbor search. Chen et al. [6] applied multi-head architecture with 3 convolutional layers to deal with each task separately. Then the missing information in the input data was recovered with the encoder-decoder structure. TrSeg [22] developed a novel semantic segmentation network by incorporating a transformer to capture multi-scale information with dependencies in an adaptive way. TransAnomaly [48] combined U-Net and transformer to capture richer temporal information and context to detect anomaly event in videos.

Conv-stem Vision Transformer: The latest research reveals that transformer achieves state-of-the-art results in multiple vision tasks. Meanwhile, the major drawback of transformer is also discovered and reported. For example, ViT is only applicable to images of fixed size owing to position embedding, while patch embedding strategy would cause the loss of local details. Although attention mechanism models long-distance dependencies quite well, it lacks the intrinsic inductive bias of CNNs. Some recent works are committed to fuse convolution blocks into classical vision transformer. Among them, CeiT [49] optimized patch embedding by adopting image-to-token (I2T) to accommodate more low-level features; ConViT [9] defined gated position self-attention and introduced inductive bias. CvT [44] attempted to use overlapped convolution to implement token embedding, so as to model the local structure (e.g. edges and lines). ConTNet [47] combined spatial convolution into transformer to create large receptive field. Xiao et al. [45] proved that early convolution could improve the robustness of ViT. Wang et al. [42] verified the conclusions of [45]from both experimental and theoretical perspectives.

Till now, most fusion architectures firstly train input images through multi-layer CNNs, and then feed them to transformer block. However, multi-layer CNNs not only increase computation cost, but also cause the loss of position information, which is critical for position information encoding in transformer.

3 Overview of our network

The overall architecture of Conv-PVT is shown in Fig. 2 above. Follow the generic CNN architecture [4], we deploy 4 layers to generate feature maps of different scales. To be more specific, the first stage is called Conv-stem, in which we attach simple lightweight convolution and ReLU function to down-sample the input image by a factor of 8. Other stages consist of patch embedding and multiple encoders. The resolution of a feature map will be reduced by 2 after propagated through patch embedding, then the feature map will be fed into the encoder to yield the token (Fig. 3).

In the first stage (Conv-stem), we use simple convolution blocks to down-sample the input image by 8, which is then fed into transformer block. In the second stage, the input is flattened into a one-dimensional vector (in other transformer stages, we use patch embedding to flatten the input) to match the dimensions of the Conv-stem. Suppose the size of the input feature map is $(H_{i-1}, W_{i-1}, C_{i-1})$, then we divide it into $\frac{H_{i-1} \times W_{i-1}}{2^2}$ patches through patch embedding, while the size of each patch is $2 \times 2 \times Ci$. Then the flattened patches is firstly projected into a specific dimension via a linear projection, followed by learnable position embedding. Then it is sent

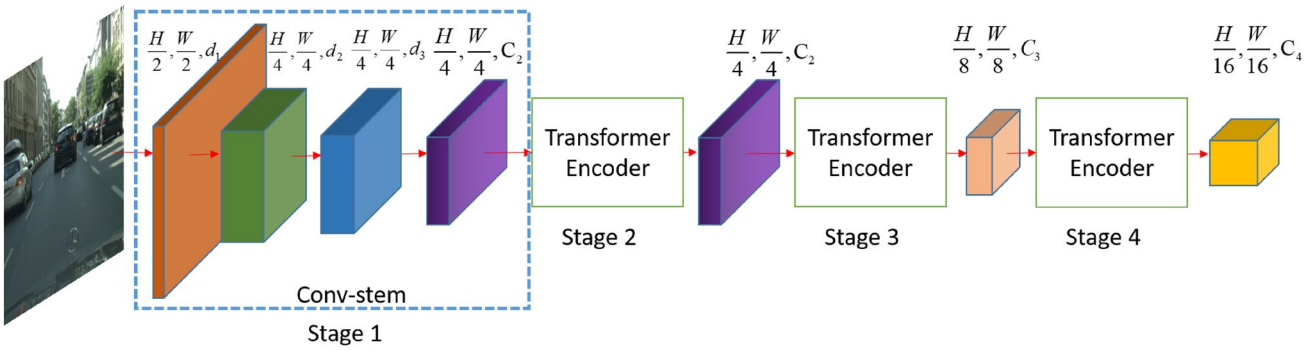

Fig. 2 The overall architecture of Conv-PVT

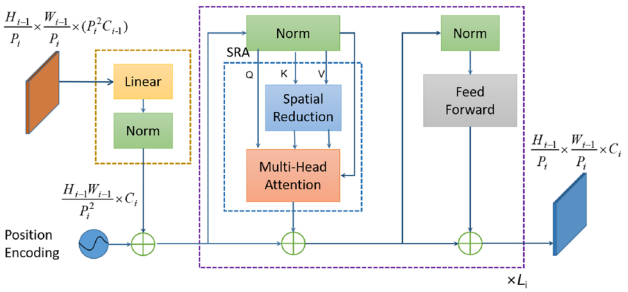

Fig. 3 The transformer encoder

to the transformer encoder. The final output is reshaped into $\frac{H_{i-1}}{2} \times \frac{W_{i-1}}{2} \times C_i$.

3.1 Conv-stem

A Conv-stem is created by stacking 3×3 and 7×7 convolutions. Then we append the Conv-stem into Batch-Norm + ReLU structure, which restricts the data in a small range [42]. The Conv-stem consists of 4 Conv + 3BN + 3ReLU + 1 Projection, and the kernel sizes and strides are (7,3,3,3) and (2,2,2,1), respectively. The Conv-stem quickly down-samples the input image (224×224) by a factor of 2 in each stage into a final size of 28×28. In the first convolution layer, we adopt the kernel size 7×7 with stride 2 and padding 3 to gain a larger receptive field. The rest of the layers use the kernel sizes (3, 3, 3) and strides (2, 2, 1). As a result, the inductive bias is implanted into transformer structure by stacking overlapped convolutions in order to capture more fine-grained local features. In vision transformer, the computational cost for multi-attention module is $O(N^2d)$, The larger the input image, the large the value of N will be. Therefore, down-sampling the input reduces the resolution of feature map (fed into transformer) while generating rich semantic information and content description.

3.2 Transformer encoder

In the last 3 stages of Conv-PVT, we use L_i encoder layers in the i^{th} stage, and each encoder layer comprises an attention block and one feed-forward block. Follow Multi-Head Attention (MHA) [41], we adopt spatial reduction attention (SRA) [43] to further reduce network scales, (the spatial scale of K (key) and V (value)). Formally,

$$SR(X) = Norm(Reshape(X, R_i)W^S) \tag{1}$$

Here, R_i denotes the reduction ratio of attention layers in the i^{th} stage. The input ($x \in \mathbb{R}^{H,W,C}$) is reshaped to the size of $\frac{HW}{R^2} \times (RC)$. $W^S \in \mathbb{R}^{(R^2C)\times C}$ is a linear projection that reduces the dimension of input to Ci. Finally we obtain $SR(X) \in \mathbb{R}^{\frac{H}{R}\times\frac{W}{R}\times C}$.

$$head_j = Attention(QW_j^Q, SR(K)W_j^K, SR(V)W_j^V) \tag{2}$$

Compared with the original transformer:

$$head_j = Attention(QW_j^Q, KW_j^K, VW_j^V) \tag{3}$$

$$SRA(Q, K, V,) = Concat(head_0, \ldots head_{N_i})W^O \tag{4}$$

where Attention (.) is expressed as:

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_{head}}})V \tag{5}$$

Contrasted with MHA, SRA reduces the computational cost by R_i^2, which endows our model with reduced memory footprint. The meaning of each variable is explained as follows:

- R_i the reduction ratio of the SRA in stage i
- K the kernel size of convolution
- S the stride of the convolution
- P the padding of the convolution
- L_i the number of encoder layers in stage i

- C_i the channel of encoder layers of stage i
- N_i the number of heads in stage i
- E_i the expansion ratio of the feed-forward layer in stage i
- d_i the i^{th} output channel of convolution

Following the design of PVT and ViTs, we also create a series of models with different scales, namely -Tiny, -Small, -Medium and -Large. The parameter settings are listed in Table 1 above.

4 Experiments

In this section, we test the effectiveness of our model by completing 3 vision tasks, including classification, detection and segmentation. Firstly, we pre-train all the proposed models on the training set and test the performance on the validation set of ImageNet-1k [10], which contains a training set of 1.28M images and a validation set of 50K images with 1k categories. Then we utilize RetinaNet [29] and Mask R-CNN [18] for objection detection and semantic FPN [26] for instance segmentation, respectively. We also compare the robustness between Conv-PVT and PVT on 3 datasets: ImageNet-A [20], ImageNet-C [20], and Stylized-ImageNet [13]. The parameters for experimental setting are explained below:

- Weight decay: An attenuation factor to accelerate convergence and prevent over-fitting.
- Momentum: A common parameter used in gradient descent process.
- Epochs: number of iterations during training.
- Gflops: Giga Floating-point Operations per Second (the lower the better)
- #Param: Number of parameters (the lower the better)
- Top-1%: An index to indicate the accuracy of prediction (the higher the better)

Image Classification: For fair comparisons, we follow the configuration of PVT [43], including random-size cropping to 224×224, random horizontal flipping [38] and mix-up [51] for data augmentation. We also employ Label-smoothing regularization [39] during training. AdamW optimizer is used with a weight decay of 5×10^{-2} and a momentum of 0.9. The initial learning rate is set to 10^{-3} and decreases following the cosine schedule [31]. All models are trained 300 epochs on 2 RTX 3090 GPUs. In the meantime, we adopt center-crop on the validation set, in which 224×224 pixels are cropped for evaluation of the recognition accuracy.

Results: We compare the performance of our model with typical CNN-based models and transformer-based models on ImageNet-1k. A comparison is made in Table 2. Here, we find an interesting pattern: Compared with PVT-Tiny, Conv-PVT-Tiny improves Top-1% accuracy by 1.4% and Conv-PVT-Small achieves the same Top-1% accuracy as PVT-Small. With the increase of network scales, Conv-PVT-Medium lags behind PVT-Medium in accuracy and Conv-PVT-Large is 1% lower than PVT-Large. The reason is that by replacing the encoder with convolution blocks in the first layer, Conv-PVT has a much faster speed of convergence in the early stage of training (in the first 100 epochs) than PVT, owing to the inductive bias introduced by Conv-stem. However, the fast down-sampling operations also bring about

Table 1 Detailed settings of Conv-PVT series

	Output Size	Layer-Name	Conv-PVT -Tiny	Conv-PVT-Small	Conv-PVT -Medium	Conv-PVT -Large
Stage1	H/2, W/2	Conv1	Kernel size = 7, Stride = 2, padding = 3, output channel = 64			
	H/4, W/4	Conv2	Kernel size = 3, Stride = 2, padding = 1, output channel = 128			
	H/8, W/8	Conv3	Kernel size = 3, Stride = 2, padding = 1, output channel = 256			
	H/8, W/8	Conv4	Kernel size = 3, stride = 1, padding = 1, output channel = 128			
Stage2		Rearrange	$C_2 = 128$			
	H/8, W/8	Encoder	$\begin{bmatrix} R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix} \times 8$
Stage3		Embedding	$P_3 = 2; C_3 = 320$			
	H/16, W/16	Encoder	$\begin{bmatrix} R_3 = 2 \\ N_3 = 5 \\ E_3 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} R_3 = 2 \\ N_3 = 5 \\ E_3 = 4 \end{bmatrix} \times 6$	$\begin{bmatrix} R_3 = 2 \\ N_3 = 5 \\ E_3 = 4 \end{bmatrix} \times 18$	$\begin{bmatrix} R_3 = 2 \\ N_3 = 5 \\ E_3 = 4 \end{bmatrix} \times 27$
Stage4		Embedding	$P_4 = 2; C_3 = 512$			
	H/32, W/32	Encoder	$\begin{bmatrix} R_4 = 2 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} R_4 = 2 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} R_4 = 2 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} R_4 = 2 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix} \times 3$

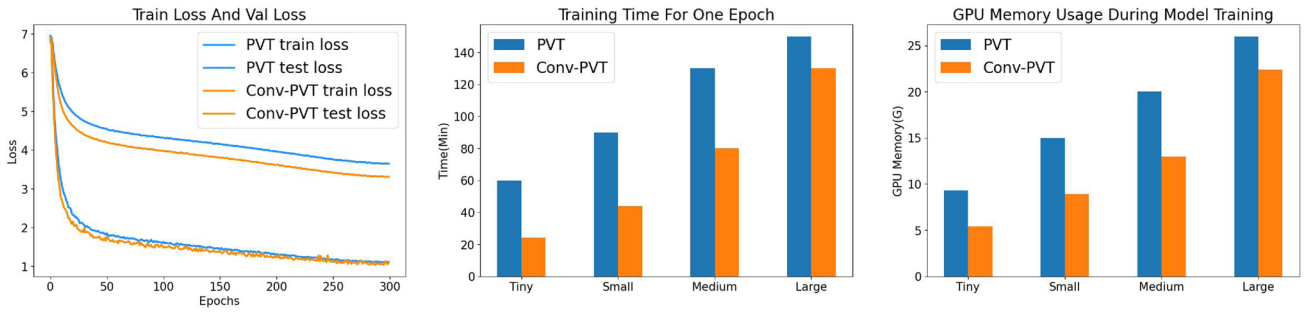

Fig. 4 The training loss, training time and GPU memory useage of PVT and conv-PVT

Table 2 Performance of image classification on ImageNet validation set. "Param" refers to the number of parameters. "GFLOPs" is calculated under the input scale of 224×224

Method	#Param (M)	GFLOPs	Top-1(%)
R18 [17]	11.7	1.8	68.5
DeiT-Tiny/16 [40]	5.7	1.3	72.2
PVT-Tiny [43]	13.2	1.9	75.1
Conv-PVT-Tiny	13	2.19	76.5
R50 [17]	25.6	4.1	74.1
X50-32x4d [46]	25.0	4.3	79.5
Deit-Small/16 [40]	22.1	4.6	79.7
PVT-Small [43]	24.5	3.8	79.8
Conv-PVT-Small	23.29	3.55	79.8
R101 [17]	44.7	7.9	79.8
X101-32x4d [46]	44.2	8	80.6
ViT-Small/16 [11]	48.8	9.9	80.8
PVT-Medium [43]	44.2	6.7	81.2
Conv-PVT-Medium	43	6.4	80.7
X101-64x4d [46]	83.5	15.6	81.5
ViT-Base/16 [11]	86.6	17.6	81.8
DeiT-Base/16 [40]	86.6	17.6	81.8
PVT-Large [43]	61.4	9.8	81.7
Conv-PVT-Large	60.8	9.8	80.7

side effect, which is the loss of the fine-grained features. While the network layer goes deeper, this effect becomes more obvious. A comparison of loss vs epochs between PVT and Conv-PVT is drawn in Fig. 4 below. The loss values of Conv-PVT decrease faster than PVT for the first 50 epochs. And the training time for Conv-PVT is obviously shorter than PVT with less memory footprint, especially for Tiny. Since the main difference among models of different sizes is the number of stacked transformer blocks. Therefore, with the increase of model size, the training time and memory usage increase accordingly, and the previous performance gaps become narrow.

Object detection: Object detection experiments are conducted on the challenging COCO benchmark [28]. All models are trained on COCO training set 2017 with 118 images and evaluated on the validation set 2017 with 5k images. We use RetinaNet [27] and Mask R-CNN [18] as our detectors. During training, we firstly use the pre-trained weights on ImageNet to initialize the backbone and Xaiver [14] to initialize the newly added layers. Our models are trained on 2 RTX 3090 GPUs and are optimized by AdamW [32] with an initial learning rate of 1×10^{-4}. Following the common setting, we adopt 1x or 3x training schedule(i.e.,12 or 36 epochs)to train RetinaNet and use 1x training schedule for Mask R-CNN. We resize the training image to the shorter side of 800 pixels, while the longer size does not exceed 1333 pixels. In the testing phase the shorter size of the input image is fixed to 800 pixels.

Result: Through Table 3, we find that Conv-PVT has completely different performances on 2 different types of detectors: Conv-PVT outshines pure convolution based models (ResNet and ResNetXt) on one-stage detector RetinaNet, but its accuracies are 2-4% lower than PVT; Conv-PVT surpasses PVT with less parameters and memory footprint (the accuracy of Conv-PVT-Tiny is 1% higher than PVT-Tiny) on two-stage detectors Mask R-CNN. Furthermore, when we extend the training time on RetinaNet, Conv-PVT achieves comparable performance with PVT. We believe the reason behind this phenomenon is that for one-stage detectors, it takes longer time to mitigate the imbalanced sample problem via fusion of the feature maps of early convolution output and transformer output. While, two-stage detectors solve such problems in the first stage.

Semantic Segmentation: We use ADE20k [55], a challenging scene parsing benchmark for semantic segmentation. ADE29k contains 150 fine-grained semantic categories, 20210 images for training, 2000 images for validation and 3352 images for testing. We evaluate our backbone by applying it to semantic FPN [24], which is a general method without dilated convolutions [37].

In the training phase, the backbone is initialized with the pre-trained weights on ImageNet, while other layers are initialized with Xavier [14]. We optimize our models with AdamW [32] where the initial learning rate is set as

Table 3 Comparison of semantic segmentation results conducted on COCO val 2017 with different backbones

Backbone	RetinaNet 1x/3x+MS							Mask R-CNN 1x						
	#Param (M)	AP	AP50	AP75	APS	APM	APL	#Param (M)	APb	APb 50	APb 75	APm	APm 50	APm 75
ResNet18 [17]	21.3	31.8/35.4	49.6/53.9	33.6/37.6	16.3/19.5	34.3/38.2	43.2/46.8	31.2	34.0	54	36.7	31.2	51.0	32.7
PVT-Tiny [43]	23.0	36.7/39.4	56.9/59.8	38.9/42.0	22.6/25.5	38.8/42.0	50.0/52.1	32.9	36.7	59.2	39.3	35.1	56.7	37.3
Conv-PVT-Tiny	22.7	33.1/40.0	51.8/60.1	34.7/42.3	18.8/26.0	35.1/42.5	44.1/52.5	32.6	37.4	60.0	49.2	35.4	56.8	37.4
ResNet50 [17]	37.7	36.3/39.0	55.3/58.4	38.6/41.8	19.3/22.4	40.0/42.8	48.8/51.6	44.2	38.0	58.6	41.4	34.4	55.1	36.7
PVT-Small [43]	34.2	40.4/42.2	61.3/62.7	43.0/45.0	25/26.2	42.9/45.2	55.7/57.2	44.1	40.4	62.9	43.8	37.8	60.1	40.3
Conv-PVT-Small	33.03	36.1/42.5	55.3/63.0	38.0/42.3	21.1/26.4	38.7/45.5	47.5/57.5	42.9	40.3	62.4	43.9	37.7	59.7	40.5
ResNet101 [17]	56.7	38.5/40.9	57.8/60.1	41.2/44.0	21.4/23.7	42.6/45.0	51.1/53.8	63.2	40.4	61.1	44.2	36.4	57.7	38.8
ResNetXt101-32x4d [46]	56.4	39.9/41.4	59.6/61.0	42.7/44.3	22.3/23.9	44.2/45.5	52.5/53.7	62.8	41.9	62.5	45.9	37.5	59.4	40.2
PVT-Medium [43]	53.9	41.9/43.2	63.1/63.8	44.3/46.1	25.0/27.3	44.9/46.3	57.6/58.9	53.9	42.0	64.4	45.6	39.0	61.6	42.1
Conv-PVT-Medium	52.7	38.4/43.0	57.5/63.5	40.8/45.8	22.3/27.0	41.5/45.9	51.3/58.5	62.7	42.3	64.3	46.2	39.3	61.7	42.3
ResNetXt101-64x4d [46]	95.5	41.0/41.8	56.9/61.5	44.0/44.4	23.9/25.2	45.2/45.4	54.0/54.6	101.9	42.8	63.8	47.3	38.4	60.6	41.3
PVT-Large [43]	71.1	42.6/43.4	63.7/63.6	45.4/46.1	25.8/26.1	46.0/46.0	58.4/59.5	81.0	42.9	65.0	46.6	39.5	61.9	42.5
Conv-PVT-Large	70.5	Out of memory						80.4	Out of memory					

Table 4 Semantic segmentation conducted on ADE20K for 80K iters with performances of different backbones. "Param" refers to number of parameters

Backbone	#Param(M)	mIoU(%)
ResNet18 [17]	15.5	32.9
PVT-Tiny [43]	17.0	35.7
Conv-PVT-Tiny	16.45	34.0
ResNet50 [17]	17.0	36.7
PVT-Small [43]	28.2	39.8
Conv-PVT-Small	26.8	37.6
ResNet-101 [17]	47.5	38.8
ResNetXt101-32x4d [46]	47.1	39.7
PVT-Medium [43]	48.0	41.6
Conv-PVT-Medium	46.49	40.2
ResNetXt101-64x4d [46]	86.4	40.2
PVT-Large [43]	86.4	42.1
Conv-PVT-Large	70.51	40.0

Table 5 The comparisions among different design of Conv-stem

Conv-stem	Kernel size	Dim	Acc
4conv+1proj	(7,3,3,3,3)	(32, 64,128, 256, 128)	67.0
4conv+1proj	(3,3,3,3,3,)	(32,64,128,256,128)	66.8
3conv+1Proj	(7,3,3,3)	(64, 128, 256,128)	65.7
3conv+1proj	(3,3,3,3)	(64, 128, 256, 128)	65.2
2conv+1proj	(7,3,3)	(64, 128,128)	63.1
2conv+1proj	(3,3,3)	(64, 128, 128)	63.0

Table 6 Semantic segmentation conducted on ADE20K for 160K iters with performances of different backbones. "Param" refers to number of parameters

Backbone	#Param (M)	mIoU(%)
PVT-Tiny [43]	17.0	36.3
Conv-PVT-Tiny	16.45	37.2
PVT-Small [43]	28.2	40.8
Conv-PVT-Small	26.8	40.8
PVT-Medium [43]	48.0	42.3
Conv-PVT-Medium	46.49	42.9
PVT-Large [43]	86.4	42.9
Conv-PVT-Large	70.51	43.0

1×10^{-4}1e-4. Suggested by [3, 23], we train our models 80K and 160K iterations respectively with the batch size of 16 on 2 RTX 3090 GPUs. The learning rate decays according to the polynomial decay scheme with a power of 0.9. We randomly crop the training image to 512×512 and resize the shorter side of the image to 512 during testing.

Results: Through Tables 4, 5, we find that Conv-PVT makes better results than ResNet and ResNetXt through 80k iterations of training on FPN, but it is still 1-2% worse than PVT. When we extend the training time to 160k, as demonstrated in Table 6, Conv-PVT outperforms PVT. This is akin to the situation of object detection using RetinaNet that it takes longer training time to fuse the outputs of convolution and transformer blocks, so that an equally long training time is needed for down-stream assignments with multi-scale feature maps.

Evaluations of Robustness: We tested the robustness of our pre-trained model on 3 datasets using 2 RTX3090 GPU: (1) ImageNet-A [20] includes 200-class subjects of ImageNet-1K, which is a dataset of adversarial filtered images of real-world; (2) ImageNet-C [19] consists of 15 types of algorithmically generated corruptions from noise, weather, and digital categories. Each type of corruption has 5 levels of severity, resulting in 75 distinct corruptions; (3) Stylized-ImageNet [13] is created by applying AdaIN style transfer to ImageNet images. After applying AdaIN style transfer, local texture cues are no longer highly predictive of the target class, while the global shape tends to be retained.

Results: As reported by [2] and shown in the Table 7, the traditional transformer has a stronger robustness on out-of-distribute samples than CNN based networks. We further validate this conclusions in this section. Especially, when we introduce CNNs into transformer, the robustness of out-of-samples reduces. The deeper the network, the greater the performance gap (for the networks of the same size).

5 Ablation study

Position Embedding: As shown in Fig. 2, the Stage 2 to Stage 4 are constructed the same way with transformer encoder. In this section, we verify the effectiveness of the position embedding by removing the it from Transformer Encoder in different layers of Conv-PVT-Tiny. Then we train it 100 epochs on ImageNet (show in Table 8). Results prove that by removing position embedding, we suffer different degrees of loss. Especially, when we remove position embedding from the 2nd layer, the accuracy decreased by 0.9%; if we remove it from both the 2nd and the 3rd layers, the accuracy decreased by 0.8%. Finally, it decreased by 1.0% without any position embedding. The function of position embedding could be described as follows: The feature map of each layer is reshaped into a two-dimensional vector, which is then flattened to form the spatial information via position embedding. Without it, the spatial information is disrupted.

Table 7 Tests of robustness on three different datasets

Model	#Param (M)	ImageNet -A	ImageNet -C	Stylized -ImageNet
PVT-Tiny	13.2	7.6	59.79	22.2
Conv-PVT-Tiny	13.0	9.2	51.3	24.7
PVT-Small	24.5	17.7	50.1	27.0
Conv-PVT-Small	23.3	17.6	45.2	27.4
PVT-Medium	44.2	24.91	46.8	27.9
Conv-PVT-Medium	43.0	20.4	42.3	28.0
PVT-Large	61.4	26.7	44.9	28.5
Conv-PVT-Large	60.8	21.4	42.9	28.8

Table 8 Experimental results for different position embedding

Position encoding	Accuracy
Pos1+Pos2+Pos3	66.90%
Pos2+Pos3	66.00%
Pos3	66.10%
None	65.90%

Different Conv-stem: In this section, we try to test different settings of Conv-PVT-Tiny to find the optimal Conv-stem. As shown in Table 5, generally, the setting in the 3rd row makes the optimal configuration. When we reduce the number of convolution block, the accuracy drops quickly. However, we do not enjoy an increase of accuracy by adding more convolution blocks. Therefore, we regard 4conv+1proj with kernel size (7,3,3,3,3) as the optimal setting for Conv-PVT-Tiny.

6 Analysis

Through the above experiments, we find that replacing the Transformer encoder with CNN has indeed shortened the training time and reduced the memory footprint, but it also sacrifices a bit accuracy for medium and large models. Because the implanted CNNs down-samples the feature maps, and outputs to the transformer block, which results in the loss of fine-grained features. Stacking more transformer and CNN blocks could not improve the overall performance, but only increases the computation load. And it is against our design intention. Meanwhile, we also find that the one-stage and two-stage detectors exhibit different performances in carrying out down-stream vision tasks. The reason is one-stage detector needs longer time to mitigate the imbalanced sample problems.

7 Engineering applications

In addition to computer vision, deep learning techniques have also been widely used in other engineering aspects. For instance, Shamshirband et al. [35] summarized the application of deep learning technology in wind and solar energy resources. Chen et al. [7] predicted discharge coefficient of streamlined weirs using coupled linear regression and gated recurrent unit. A spatiotemporal model called KL-MLP-LSTM was constructed to estimate temperature distributions [12]. Chen et al. [5] and Afan et al. [1] forecast the rainfall and model fluctuations of groundwater level respectively using LSTM. Despite novel, the above methods all depend on traditional network architectures (e.g. linear regression, LSTM). Transformer is a new architecture, which excels in global context information modelling. And the improved Transformers will certainly bring significant performance gain in computational efficiency and accuracy in various engineering fields in near future.

8 Conclusions

The rising trend of Transformer has been witnessed in both NLP and vision domain. However, Transformer occupies a large volume of memory footprint. For example, 24G is the maximum capacity of an ordinary consumer graphics card, but PVT-Large needs 4×V100 (32G for each) graphics cards for training, where only well-equipped labs could afford it. Motivated by recent works on combining the CNNs and transformer for vision tasks, we propose Conv-PVT by introducing CNNs into transformer structure. Our advised network reduces the memory footprint by 40%, and shortens the training time by 60% respectively. In addition, Conv-PVT needs only 2×3090 (24G for each) for training, which makes it more affordable to normal research institutions. We particularly highlight Conv-PVT-Tiny, which is not only lightweight than PVT-Tiny, but also achieves higher accuracy in multiple downstream vision tasks. Practically, it could replace PVT-Tiny in real-time applications, including intelligent surveillance, autonomous driving and military etc. We hope our work could open a new path for future work on the acceleration of transformer based models.

In the meantime, there are still some limitations in our proposed architecture. We notice that with the increase of the network scales, our model begins to lag behind PVT in accuracy (though it is still better than ResNet and ResNeXt), and the robustness of the model is also weakened by the convolution blocks. The reason is that Conv-PVT quickly down-samples the input images through 2 convolution layers causing information loss. In the future, we will be more committed to investigating more optimized fusion architecture with higher accuracy and robustness.

Data Availability Statement All data included in this study are available upon request by contact with the corresponding author.

Declarations

Declaration of Interest Statement I declare I have no financial support and personal relationships with other people or organizations that can inappropriately influence our work. There is no professional or other personal interest of any nature or kind in any products, services, or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled.

References

1. Afan HA, Ibrahem Ahmed Osman A, Essam Y et al (2021) Modeling the fluctuations of groundwater level by employing ensemble deep learning techniques. Eng Appl Comput Fluid Mech 15(1):1420–1439
2. Bai Y, Mei J, Yuille A et al (2021) Are transformers more robust than CNNs? Adv Neural Inf Process Syst 34:2
3. Carion N, Massa F, Synnaeve G, et al (2020) End-to-end object detection with transformers. In: European Conference on Computer Vision, pp 213–229
4. Chaudhari P, Agrawal H, Kotecha K (2020) Data augmentation using MG-GAN for improved cancer classification on gene expression data. Soft Comput 24(15):11381–11391. https://doi.org/10.1007/s00500-019-04602-2
5. Chen C, Zhang Q, Kashani MH et al (2022) Forecast of rainfall distribution based on fixed sliding window long short-term memory. Eng Appl Comput Fluid Mech 16(1):248–261
6. Chen H (2021) Pre-trained image processing transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 12,299–12,310
7. Chen W, Sharifrazi D, Liang G et al (2022) Accurate discharge coefficient prediction of streamlined weirs by coupling linear regression and deep convolutional gated recurrent unit. Eng Appl Comput Fluid Mech 16(1):965–976
8. Chollet F (2017) Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1251–1258
9. d'Ascoli S, Touvron H, Leavitt M, et al (2021) ConViT: Improving vision transformers with soft convolutional inductive biases. In: ICML, vol 2, 3
10. Deng J, Dong W, Socher R, et al (2009) Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, pp 248–255
11. Dosovitskiy A, Beyer L, Kolesnikov A, et al (2021) An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR, vol 1, 2, 3, 4, 5, 10. p 13
12. Fan Y, Xu K, Wu H et al (2020) Spatiotemporal modeling for nonlinear distributed thermal processes based on kl decomposition, mlp and lstm network. IEEE Access 8:25111–25121
13. Geirhos R, Rubisch P, Michaelis C, et al (2019) ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In: International Conference on Learning Representations
14. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the

thirteenth international conference on artificial intelligence and statistics, pp 249–256
15. Goyal P (2017) Accurate, large minibatch sgd: Training imagenet in 1 hour. ArXiv Prepr ArXiv170602677
16. Guo MH, Cai JX, Liu ZN et al (2021) PCT: Point cloud transformer. Comput Vis Media 7(2):187–199
17. He K, Zhang X, Ren S, et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
18. He K, Gkioxari G, Dollár P, et al (2017) Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 2961–2969
19. Hendrycks D, Dietterich T (2019) Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In: Proceedings of the International Conference on Learning Representations
20. Hendrycks D, Zhao K, Basart S, et al (2021) Natural Adversarial Examples. CVPR
21. Howard AG, Zhu M, Chen B, et al (1704) Mobilenets: efficient convolutional neural networks for mobile vision applications. CoRR 2, 4, 5:6
22. Jin Y, Han D, Ko H (2021) TrSeg: transformer for semantic segmentation. Pattern Recogn Lett 148:29–35. https://doi.org/10.1016/j.patrec.2021.04.024
23. K. He PDG. Gkioxari, Girshick R (2017) Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 2961–2969
24. Kirillov A, Girshick R, He K, et al (2019) Panoptic feature pyramid networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 6399–6408
25. Krizhevsky A, Sutskever I, Hinton G (2012) Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 25:1097–1105
26. Lin T, Dollár P, Girshick R, et al (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2117–2125
27. Lin T, Goyal P, Girshick R, et al (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp 2980–2988
28. Lin TY (2014) Microsoft coco: common objects in context. In: European conference on computer vision, pp 740–755
29. Lin TY, Goyal P, Girshick R, et al (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp 2980–2988
30. Liu Z, Lin Y, Cao Y, et al (2021) Swin transformer: hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). p 10,012–10,022
31. Loshchilov I, Hutter F (2017) SGDR: stochastic gradient descent with warm restarts. In: Proceedings of the Internal Conference on Learning Representations 2017
32. Loshchilov I, Hutter F (2019) Decoupled weight decay regularization. In: ICLR, vol 1, 3. p 5
33. Radford A, Narasimhan K, Salimans T, et al (2018) Improving language understanding by generative pre-training
34. Raghu M, Unterthiner T, Kornblith S, et al (2021) Do vision transformers see like convolutional neural networks? In: Thirty-Fifth Conference on Neural Information Processing Systems
35. Shamshirband S, Rabczuk T, Chau KW (2019) A survey of deep learning techniques: application in wind and solar energy resources. IEEE Access 7:164,650–164,666
36. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: ICLR
37. Sun P (2021) 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 14,454–14,463
38. Szegedy C (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
39. Szegedy C, Vanhoucke V, Ioffe S, et al (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2818–2826
40. Touvron H, Cord M, Douze M, et al (2021) Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning. PMLR, pp 10,347–10,357
41. Vaswani A (2017) Attention is all you need. In: Advances in neural information processing systems. p 5998–6008
42. Wang P (2021) Scaled relu matters for training vision transformers. ArXiv Prepr ArXiv210903810
43. Wang W, Xie E, Li X, et al (2021) Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: ICCV, vol 3
44. Wu H, Xiao B, Codella N, et al (2021) CvT: Introducing convolutions to vision transformers. In: ICCV, vol 3
45. Xiao T, Dollar P, Singh M et al (2021) Early convolutions help transformers see better. Adv Neural Inf Process Syst 2:2
46. Xie S, Girshick R, Dollár P, et al (2017) Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1492–1500
47. Yan H, Li Z, Li W, et al (2021) ConTNet: Why not use convolution and transformer at the same time? In: ArXiv210413497 Cs. http://arxiv.org/abs/2104.13497
48. Yuan H, Cai Z, Zhou H, et al (2021) TransAnomaly: Video Anomaly Detection Using Video Vision Transformer. IEEE Access 9:123,977–123,986. https://doi.org/10.1109/ACCESS.2021.3109102.
49. Yuan K, Guo S, Liu Z, et al (2021) Incorporating convolution designs into visual transformers. In: ICCV, vol 3
50. Yuan L, Chen Y, Wang T, et al (2021) Tokens-to-Token ViT: Training Vision Transformers From Scratch on ImageNet. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). p 558–567
51. Zhang H, Cissé M, Dauphin YN, et al (2018) Mixup: Beyond empirical risk minimization. In: ICLR
52. Zhang P (2021) Multi-Scale Vision Longformer: A New Vision Transformer for High-Resolution Image Encoding. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV, pp 2998–3008
53. Zhang X, Zhou X, Lin M, et al (2018) Shufflenet: an extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 6848–6856
54. Zhang Y (2021) Vidtr: Video transformer without convolutions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 13,577–13,587
55. Zhou B, Zhao H, Puig X, et al (2017) Scene parsing through ade20k dataset. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 633–641