# TFormer: A Transmission-Friendly ViT Model for IoT Devices

Zhichao Lu, Chuntao Ding, Felix Juefei-Xu, *Member, IEEE,* Vishnu Naresh Boddeti, *Member, IEEE,* Shangguang Wang, *Senior Member, IEEE,* Yun Yang, *Senior Member, IEEE,*

**Abstract**—Deploying high-performance vision transformer (ViT) models on ubiquitous Internet of Things (IoT) devices to provide high-quality vision services will revolutionize the way we live, work, and interact with the world. Due to the contradiction between the limited resources of IoT devices and resource-intensive ViT models, the use of cloud servers to assist ViT model training has become mainstream. However, due to the larger number of parameters and floating-point operations (FLOPs) of the existing ViT models, the model parameters transmitted by cloud servers are large and difficult to run on resource-constrained IoT devices. To this end, this paper proposes a transmission-friendly ViT model, TFormer, for deployment on resource-constrained IoT devices with the assistance of a cloud server. The high performance and small number of model parameters and FLOPs of TFormer are attributed to the proposed hybrid layer and the proposed partially connected feed-forward network (PCS-FFN). The hybrid layer consists of nonlearnable modules and a pointwise convolution, which can obtain multitype and multiscale features with only a few parameters and FLOPs to improve the TFormer performance. The PCS-FFN adopts group convolution to reduce the number of parameters. The key idea of this paper is to propose TFormer with few model parameters and FLOPs to facilitate applications running on resource-constrained IoT devices to benefit from the high performance of the ViT models. Experimental results on the ImageNet-1K, MS COCO, and ADE20K datasets for image classification, object detection, and semantic segmentation tasks demonstrate that the proposed model outperforms other state-of-the-art models. Specifically, TFormer-S achieves 5% higher accuracy on ImageNet-1K than ResNet18 with 1.4× fewer parameters and FLOPs.

**Index Terms**—Internet of Things, cloud computing, cloud-assisted, vision transformer.

## 1 Introduction

The International Data Corporation predicts that by 2025, there will be 41.6 billion connected Internet of Things (IoT) devices [1]. Additionally, the recently proposed vision transformer (ViT) models, with the support of large datasets, have crushed the convolutional neural network models that have dominated for many years in multifarious vision tasks, such as image classification [2], [3], object detection [4], [5], and semantic segmentation [6], [7]. Deploying high-performance ViT models on ubiquitous IoT devices to provide high-quality vision services has attracted great attention from both industry and academia.

However, since IoT devices are resource-constrained (e.g., limited storage and computing resources), it is difficult to provide sufficient resources for training resource-intensive ViT models. Therefore, it has become mainstream to assist model training with the help of cloud/edge servers [8]–[13]. In general, the cloud server first trains the ViT model and then sends it to the IoT device for deployment and updating, as illustrated in Fig. 1. The ViT model contains a large number of parameters (e.g., ViT [4] contains 41 million parameters), which leads to transmitting a large number of model parameters when the cloud server assists in deploying and updating the model. In particular, in the construction of the smart city, when a cloud server assists thousands of IoT devices, the transmission of a large number of model parameters will lead to heavy network load and difficulty in online deployment and updating of

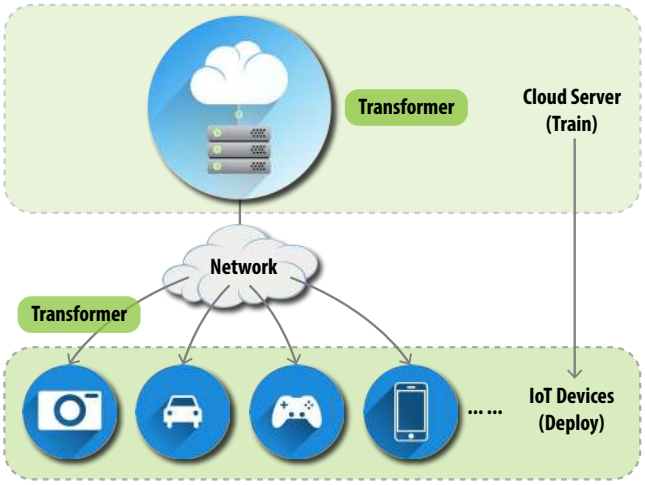



Fig. 1: Overview of system architecture.

- *Zhichao Lu is with the School of Software Engineering, Sun Yat-sen University, Zhuhai 519082, China. E-mail: luzhichaocn@gmail.com.*
- *Chuntao Ding is with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China. E-mail: chtding@bjtu.edu.cn.*
- *Felix Juefei-Xu is with Meta AI, New York, NY 10001 and New York University, New York, NY 10012, USA. This work is done prior to joining Meta AI. E-mail: felixu@meta.com, juefei.xu@nyu.edu.*
- *Vishnu N. Boddeti is with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA. E-mail: vishnu@msu.edu.*
- *Shangguang Wang is with the Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China 100876. E-mail: sgwang@bupt.edu.cn.*
- *Yun Yang is with the Department of Computing Technologies, Swinburne University of Technology, Melbourne, Australia. E-mail: yyang@swin.edu.au.*

*(Corresponding author: Chuntao Ding.)*

ViT models. In addition, existing ViT models contain a large number of floating-point operations (FLOPs), making them difficult to deploy on resource-constrained IoT devices. Therefore, to facilitate the cloud server to assist IoT devices in deploying and updating the model, it is necessary to reduce the number of model parameters and FLOPs, especially in today's Internet of Everything era. To this end, this paper aims to implement a transmission-friendly ViT model for IoT devices to reduce the number of model parameters and FLOPs while improving model performance.

First, this paper proposes a hybrid layer consisting of nonlearnable (NL) modules and a pointwise convolution to replace the multihead attention (MHA) of the standard ViT. The NL module consists of maximum (max) pooling and average (avg) pooling. Additionally, the NL module can easily and parallelly extract multitype and multiscale features to improve the transformer performance, e.g., using $3 \times 3$, $5 \times 5$, or $7 \times 7$ max/avg-pooling. The pointwise convolution consists of $1 \times 1$ convolutions with only a few learnable parameters. Therefore, replacing the MHA with the proposed hybrid layer can greatly reduce the number of model parameters and FLOPs.

Then, this paper introduces the group convolution for channel sparse connection to reduce the number of parameters in the feedforward network of the ViT. Group convolution can significantly reduce the number of parameters by ensuring that each convolution operates only on the corresponding group of input channels. Motivated by [14], we introduce the channel shuffle operation to allow group convolution to obtain input data from different groups; that is, the input and output channels can be fully related. We also present a tradeoff between the number of model parameters and model performance analyzed from the experimental results.

By introducing the hybrid layer and group convolution into the existing ViT, we propose a Transmission-Friendly vision Transformer (TFormer), which has fewer parameters and FLOPs while also achieving higher performance. It can be deployed on a large scale on IoT devices with cloud-assisted training. Since our TFormer is similar in structure to the existing ViT, we call TFormer a kind of ViT.

Finally, this paper conducts extensive experiments on the ImageNet-1K, MS COCO, and ADE20K datasets, and the experimental results show that our proposed TFormer achieves strong performance on the recognition tasks of image classification, object detection, and semantic segmentation. For example, when the number of model parameters is similar, our proposed TFormer achieves 41.2 average precision (AP) on the MS COCO dataset, which surpasses the previous state-of-the-art result by +2.3 AP. ADE20K semantic segmentation obtains 41.8 mean intersection over union (mIoU), an improvement of +2.3 mIoU over the previous state-of-the-art results. In addition, we also demonstrate the advantages of TFormer in the number of model parameters and FLOPs, making it more suitable for deployment on resource-constrained IoT devices.

In summary, our main contributions are as follows:

– This paper presents a new pathway towards efficient ViT models for IoT devices, comprising a novel hybrid layer and a novel feedforward network with group-wise connections.

– The key component of the proposed hybrid layer is the nonlearnable module with multiple non-parametric operations (i.e., pooling operators with various kernel sizes) in parallel to extract multitype and multiscale features, while at the same time reducing the model parameters and FLOPs substantially.

– Extensive experiments conducted on the ImageNet, MS COCO, and ADE20K datasets verify the effectiveness of the proposed method over classification, object detection, and semantic segmentation tasks.

The remainder of the paper is organized as follows. Section 2 reviews the related work on cloud-assisted approaches and ViT models. Section 3 describes the proposed TFormer model in detail. Section 4 presents our evaluation results, and Section 5 concludes the paper.

## 2 Related Work

In this section, we introduce the cloud-assisted approach and the ViT models that are most relevant to this paper.

### 2.1 Cloud-assisted Approach

Due to the limited resources of IoT devices, it is difficult to provide sufficient computing and storage resources for training high-performance deep neural network models (e.g., deep convolutional neural networks and transformers). To this end, it has become the mainstream to first use well-resourced cloud/edge servers to assist in training high-performance models [15]–[23]. For example, inspired by [24], Ding et al. [16], [25] proposed training a complex neural network model on a cloud server and a simple neural network model on an edge server and improving the performance of the latter by sharing some layer parameters of the complex neural network model with the simple neural network model. Teerapittayanon et al. [17] proposed a distributed deep neural network architecture consisting of cloud servers, edge servers and IoT devices. Similarly, Kang et al. [26] proposed dividing the neural network model into two parts, which are run on cloud servers and IoT devices.

Many studies have also made innovative contributions to adapting IoT devices. For example, to adapt to the dynamically changing available resources of IoT devices, Han et al. [27] proposed deploying multiple model variants on IoT devices. Fang et al. [28] proposed making multiple model variants share parameters to save the limited storage resources of the IoT device. Additionally, to reduce the number of model parameters when the cloud server assists in training the IoT neural network model, many researchers proposed using model compression [29] or knowledge distillation [30] to reduce the amount of model parameter transmission. For example, Laskaridis et al. [31] proposed using a model compression technique to reduce the number of model parameters during the interaction between the cloud server and the IoT device.

Different from the above methods, our method focuses on reducing the number of model parameters and floating-point operations by analyzing the components of the model in detail and introduces pooling techniques and group

convolutions to achieve this goal. Model compression techniques can further reduce the number of model parameters based on our method. The study of adapting IoT devices [27], [28] can also be built on our method.

### 2.2 Vision Transformers

Considering the great success of the transformer [32] in the natural language processing field, the application of the transformer architecture to the vision field has attracted the attention of a large number of scholars and achieved attractive results [2], [3], [5], [33]–[35]. For example, Dosovitskiy et al. [2] pioneered the standard transformer to process images directly. They split the image into patches and provide a sequence of linear embeddings of these patches as input to the transformer. Liu et al. [5] proposed a hierarchical transformer (swin transformer) to support large-scale variations of visual entities and high-resolution pixels in images. To improve the convergence rate of the transformer, Touvron et al. [3] proposed a teacher-student strategy for the transformer. Wu et al. [33] introduced convolution in the standard transformer to improve the performance of the transformer so that the transformer containing the convolution affords both the advantages of the convolution and the standard transformer. Similarly, Yuan et al. [34] also introduced convolution in the standard transformer to improve the transformer's performance in vision tasks.

Accordingly, existing vision transformers achieve high performance by leveraging the multihead attention module to capture the global information of data. However, due to the quadratic complexity of the multihead attention module, it is difficult for existing ViT models to be widely deployed in resource-constrained IoT devices. In addition, the feedforward network module included in the existing ViT model contains a large number of parameters, which makes it necessary to transmit a large number of model parameters during cloud-assisted deployment and updating. This will hinder the wide deployment of the ViT model in IoT devices in the era of the Internet of Everything, where bandwidth resources are tight. To this end, we analyze the components of the ViT model in detail from the perspective of cloud-assisted deployment to reduce the number of model parameters and floating-point operations, as well as maintain the performance of the model.

Existing efforts toward improving the efficiency of ViT models can be broadly classified into three categories. The first group of methods focuses on reducing the complexity of the attention module by imposing the locality of input images adaptively [36], [37]. The second group applies pruning methods to remove unimportant components (e.g., partial channels) [38] or inputs (i.e., patches) [39] to a ViT model. The third group of methods uses neural architecture search (NAS) techniques to design efficient ViT models by optimizing architectural hyperparameters, such as channels and depth [40], [41]. On one hand, the improvements in model efficiency provided by adaptive attention mechanisms are still insufficient for tasks with high-resolution imagery (e.g., object detection, segmentation, etc.). On the other hand, despite the promising results, both pruning and NAS-based approaches are computationally expensive, requiring days to weeks on a cluster of GPUs to execute the methods.

## 3 Design of the Proposed Approach

In this section, we first introduce the overview of the proposed framework and motivation. We then provide the detailed design of our proposed hybrid layer and partially connected and shuffled feedforward network (PCS-FFN).

### 3.1 Overview

TABLE 1: Specific design details of three TFormer variants.

| Stage | #Tokens | Specifications | TFormer S \| M \| L |
|---|---|---|---|
| 1 | $\frac{H}{4} \times \frac{W}{4}$ | Patch size | 7×7, stride 4 |
| | | Embed. dim. | 64 |
| | | FFN | ratio 4, groups 2 |
| | | #Layers | 2 \| 4 \| 6 |
| 2 | $\frac{H}{8} \times \frac{W}{8}$ | Patch size | 3×3, stride 2 |
| | | Embed. dim. | 128 |
| | | FFN | ratio 4, groups 2 |
| | | #Layers | 2 \| 4 \| 6 |
| 3 | $\frac{H}{16} \times \frac{W}{16}$ | Patch size | 3×3, stride 2 |
| | | Embed. dim. | 320 |
| | | FFN | ratio 4, groups 2 |
| | | #Layers | 6 \| 12 \| 18 |
| 4 | $\frac{H}{32} \times \frac{W}{32}$ | Patch size | 3×3, stride 2 |
| | | Embed. dim. | 512 |
| | | FFN | ratio 4, groups 2 |
| | | #Layers | 2 \| 4 \| 6 |
| Parameters (M) | | | 8 \| 14 \| 20 |
| Multi-Adds (G) | | | 1.2 \| 2.2 \| 3.2 |

The overall framework first trains a vision transformer (ViT) model on the cloud server. Then, the cloud server sends the trained model to IoT devices to provide a variety of convenient services, such as object detection and segmentation services. See Fig. 2 for a pictorial overview.

The goal of this paper is to develop a transmission-friendly ViT model while ensuring its performance. To fulfill this goal, we first propose a compact yet effective ViT model, dubbed TFormer, tailored for IoT applications. As depicted in Fig. 2 (top-right), the main computational block of TFormer comprises a *hybrid* layer and a *partially connected and shuffled* feedforward network (PCS-FFN), replacing the multihead attention (MHA) layer and the standard FFN in existing ViTs, respectively. The design principles of the hybrid layer and PCS-FFN are carefully explained and empirically validated in Section 3.3 and Section 3.4, respectively. Subsequently, we develop three variants of TFormer with an increasing model capacity, i.e., TFormer-S, TFormer-M, and TFormer-L. The specific configuration details of the three variants are provided in Table 1.

### 3.2 Motivation

Generally, there are two aspects to measure whether a model is suitable for running on resource-constrained IoT devices,

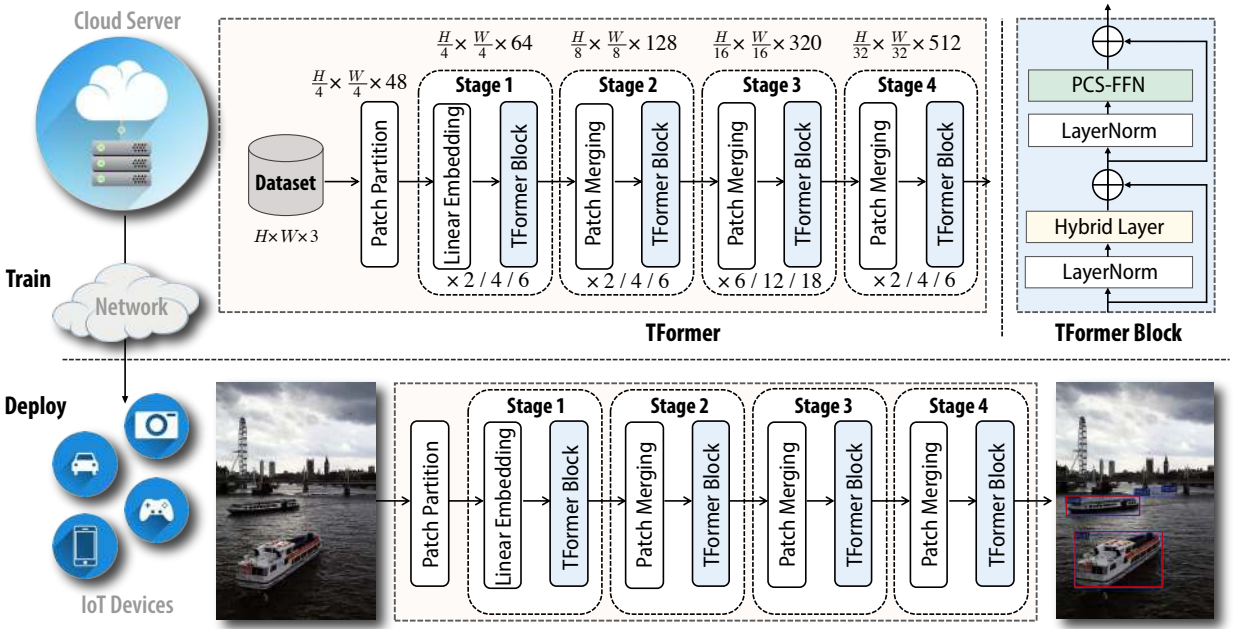



Fig. 2: Overview of the proposed framework.

namely, the number of model parameters and the number of floating-point operations (FLOPs) required to run the model. To be consistent with prior work, we consider the number of floating-point operations for multiplication and addition together as one FLOP [42].

On the one hand, the number of model parameters affects the deployment of the model in two ways: (i) the quantity of data required to be transmitted between cloud and IoT devices and (ii) the size of the model. First, we aim to train a ViT on the cloud server, which is then sent to IoT devices for deployment. In the Internet of Everything era, considering the online deployment and updating of models on hundreds of millions of IoT devices, it is necessary to reduce the quantity of data sent by the cloud server to IoT devices. Second, the number of model parameters determines both the storage space and memory resources of the IoT device that will be occupied by the model.

On the other hand, the number of FLOPs signifies the computational complexity of a ViT model, where a higher value of FLOPs typically results in higher power consumption and longer inference latency [43], [44]. Particularly, IoT devices are mostly constrained by resources, such that the level of computational complexity (FLOPs) of a ViT model is especially important.

Therefore, to facilitate cloud-assisted deployment and updating of models on IoT devices, it is necessary to reduce the number of parameters and the number of FLOPs of the ViT model *simultaneously*.

### 3.3 Design of Hybrid Layer

**Preliminaries.** The multihead attention (MHA) is one of the cornerstones in existing ViT models [2], [5], [32], [35], [45]. Undoubtedly, MHA is important for model performance owing to its ability to extract nonlocal dependencies. Nevertheless, the steep computational overhead has greatly restricted its application to resource-constrained hardware, such as IoT devices.

Specifically, let us consider an input $\mathbf{X} \in \mathbb{R}^{N\times D}$, where $N$ is the size of the input (i.e., number of pixels) and $D$ is the embedding dimension (i.e., number of channels). Assuming the number of attention heads is $h$, then for each head $i$, the input $\mathbf{X}$ is embedded into a *query* ($\mathbf{Q}_i$), *key* ($\mathbf{K}_i$), and *value* ($\mathbf{V}_i$), as follows:

$$\mathbf{Q}_i = \mathbf{X}\mathbf{W}_i^Q, \quad \mathbf{K}_i = \mathbf{X}\mathbf{W}_i^K, \quad \mathbf{V}_i = \mathbf{X}\mathbf{W}_i^V, \tag{1}$$

where $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, and $\mathbf{W}_i^V \in \mathbb{R}^{D\times D/h}$ are learnable parameters for $i = 1, \ldots, h$. The attention for the $i$th head is then computed as follows:

$$\text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = softmax\Big(\frac{\mathbf{Q}_i\mathbf{K}_i^\intercal}{\sqrt{\frac{D}{h}}}\Big)\mathbf{V}_i \ . \tag{2}$$

Then, the outputs from all heads are concatenated along the channel dimension:

$$[\text{head}_1, \ldots, \text{head}_h] = \text{Concat}\big(\text{Attention}(\mathbf{Q}_1, \mathbf{K}_1, \mathbf{V}_1), \ldots, \text{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h)\big) \tag{3}$$

Finally, the concatenated outputs are projected via another learnable parameter $\mathbf{W} \in \mathbb{R}^{D\times D}$, yielding the final outputs of the MHA as follows:

$$\text{MHA}(\mathbf{X}) = [\text{head}_1, \ldots, \text{head}_h]\mathbf{W} \tag{4}$$

Accordingly, the number of parameters of MHA can be derived as:

$$P_{\text{MHA}} = \underbrace{3\sum_{i=1}^{h} D \times \frac{D}{h}}_{\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V} + \underbrace{D \times D}_{\mathbf{W}} = 4D^2, \tag{5}$$

which is *quadratic* to the embedding dimension $D$. Considering typical embedding dimension values (i.e., $D \in \{128, 320, 512\}$) used in existing ViTs [2], [3], the resulting number of parameters can be prohibitive to transmit between cloud and IoT devices. Similarly, we can derive the number of FLOPs of MHA as follows:

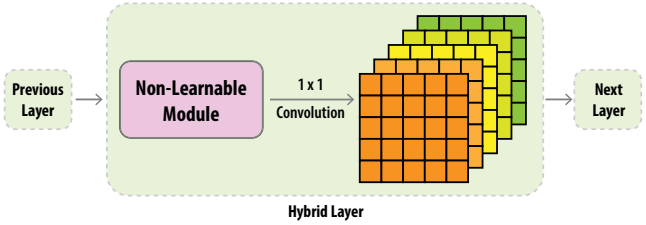


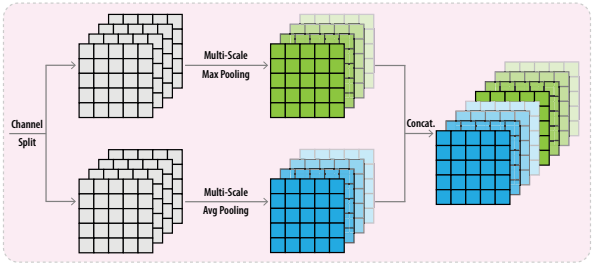

Fig. 3: **Hybrid layer:** It first uses the nonlearnable module to extract multitype and multiscale features and then uses pointwise ($1 \times 1$) convolution to enable these features to establish information interactions between channels.

$$F_{\text{MHA}} = \underbrace{3ND\frac{D}{d}}_{\text{Eq 1}} + \underbrace{N\frac{D}{h}N + NN\frac{D}{h}}_{\text{Eq 2-3}} + \underbrace{NDD}_{\text{Eq 4}}. \tag{6}$$

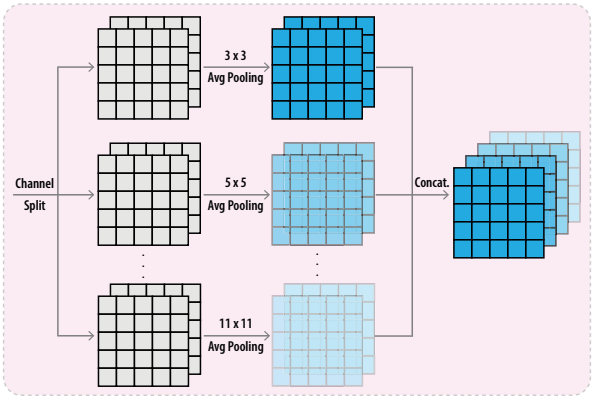
Since the input size is typically much greater than the embedding dimension (i.e., $N \gg D$), $F_{\text{MHA}}$ is approximately *quadratic* to the input size $N$. Given the input sizes of the considered tasks (e.g., $1200\times800$ for object detection), one computation of MHA can already be too expensive for IoT devices.

**Design principles.** To this end, we aim to propose a novel layer to replace MHA for reducing both the number of model parameters and FLOPs while maintaining similar model performance. As explained by the original authors [32], the role of MHA is to enable ViT models to jointly focus on information from different representation subspaces at different locations. From this point of view, it can be equivalent to using multiple types of filters to extract diverse features. Under this assumption, we propose a hybrid layer that contains only a few learnable parameters but can extract multitype and multiscale features. As depicted in Fig. 3, the proposed hybrid layer comprises a nonlearnable (NL) module and a pointwise (i.e., $1\times 1$) convolution. In this hybrid layer, the NL module is used to extract diverse spatial information, while the pointwise convolution is learned to recombine the multitype and multiscale information.

As depicted in Fig. 4a, our NL module comprises a max-pooling operation and an avg-pooling operation. The max-pooling operation calculates the maximum value of feature patches, which is known to be invariant to a small amount of translation. In parallel, the avg-pooling operation calculates the mean value for feature patches. Furthermore, we allow each pooling operation to have multiple kernel size settings in parallel to capture multiscale features, as depicted in Fig. 4b. We set the padding accordingly to maintain the same spatial size of inputs. More specifically, let's consider that there are one input and one output feature map. Then the size of the extracted output feature map is calculated as $N = [(W-K+2P)/S]+1$, where $W$ is the size of the input feature map; $K$ is the pooling size (e.g., $3 \times 3$, $5 \times 5$, etc.), $P$ is the padding size; $S$ is the stride. To maintain the same sizes between input $W$ and output $N$ feature maps under different pooling size $K$, we first set the stride $S$ to one then adjust the padding $P$ accordingly. For instance, we set $P$ to 1 and 2 for pooling sizes of $3 \times 3$ and $5 \times 5$ respectively. Technically, one may have as many NL operations in parallel as one wishes, and we reserve this part of the work as in our future studies.




(a) Nonlearnable modules




(b) Multiscale average pooling

Fig. 4: (a) Our nonlearnable modules comprise nonparametric pooling operations, i.e., avg and max poolings. (b) We allow each pooling operation to have multiple kernel size settings in parallel to capture multiscale features.

Note that we split the inputs along the channel dimension for each NL operator and for each kernel size within that operator to further reduce the computations. Specifically, assuming that the number of input channels is $D$ and the number of considered kernel sizes is $m$, such a channel splitting operation will maintain the same number of output channels (as inputs) of $D$, instead of $2 * D * m$ for the case without channel splitting. Then, the number of learnable parameters of the proposed hybrid layer is simply the number of parameters of the pointwise convolution, i.e., $P_{\text{NL Layer}} = D^2$ assuming an equal number of input and output channels. Hence, the ratio of the number of parameters between the MHA and our hybrid layer is as follows:

$$R_{\text{P}} = \frac{P_{\text{MHA}}}{P_{\text{NL Layer}}} = \frac{4D^2}{D^2} = 4. \tag{7}$$

Apparently, compared to MHA, our hybrid layer leads to a $4\times$ reduction in the number of parameters under the assumption of equal embedding dimensions and channels.

The number of FLOPs of the proposed hybrid layer can be computed as:

$$F_{\text{NL Layer}} = \underbrace{DNk^2}_{\text{NL modules}} + \underbrace{D^2N}_{1\times1\text{ Conv.}}, \tag{8}$$

where $k$ is the kernel size, $N$ is the input size (i.e., number of pixels), and $D$ is the number of input/output channels. Since the kernel size is typically much smaller than the number of channels (i.e., $k \ll D$), $F_{\text{NL Layer}}$ is approximately

quadratic to the number of channels $D$. Thus, the ratio of the number of FLOPs between MHA and our hybrid layer can be approximated as follows:

$$R_{\text{F}} = \frac{F_{\text{MHA}}}{F_{\text{NL Layer}}} \sim \frac{N^2}{D^2}. \tag{9}$$

Given the vision tasks considered in this work (e.g., object detection, segmentation, etc.), the input size (e.g., $N = 1200 \times 800$ for object detection) is much greater than the number of channels (e.g., $D = 512$), and the resulting FLOPs ratio is expected to be much greater than 1 (i.e., $R_{\text{F}} \gg 1$). Hence, our hybrid layer also leads to significant savings in the number of FLOPs compared to MHA.

TABLE 2: Ablation studies of our nonlearnable modules. All variants have similar number of parameters and FLOPs.

| Non-learnable operator | | Scale | | | | | ImageNet Top-1 Acc |
|---|---|---|---|---|---|---|---|
| Avg. Pool | Max Pool | 3×3 | 5×5 | 7×7 | 9×9 | 11×11 | |
| ✓ | | ✓ | | | | | 77.2% |
| ✓ | | | ✓ | | | | 77.2% |
| ✓ | | | | ✓ | | | 77.1% |
| ✓ | | | | | ✓ | | 76.8% |
| ✓ | | ✓ | ✓ | | | | 77.2% |
| ✓ | | ✓ | ✓ | ✓ | | | 77.3% |
| ✓ | | ✓ | ✓ | ✓ | ✓ | | 77.5% |
| ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | 77.7% |
| ✓ | ✓ | ✓ | | | | | 77.6% |
| ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 78.0% |

**Effectiveness.** Next, we provide empirical validations of our hybrid layer. Specifically, we consider the image classification task on the ImageNet-1K dataset [46]; we replace the MHA layer with our hybrid layer in ViTs [2]; and compare different variants of our hybrid layer. Experimental results are summarized in Table 2. We observe that both adding more types of operations and adding more scales (i.e., kernel sizes) are beneficial to the model performance without additional learnable parameters. These results provide the empirical basis for the design of our hybrid layer.

### 3.4 Design of PCS-FFN

**Preliminaries.** In addition to MHA, the other main computational bottleneck in existing ViT models is the feedforward network (FFN), which is usually placed immediately after the MHA layer (see Fig. 2 top-right). An FFN comprises two fully connected (linear) layers with an activation in between, as shown in Fig. 5a. For an input $\mathbf{X}_{\text{in}} \in \mathbb{R}^{N \times D}$, the first linear layer expands the number of channels to a ratio of $r$, yielding an intermediate output $\mathbf{X}_{\text{inter}} \in \mathbb{R}^{N \times rD}$. Then, the second linear layer compresses channels back to their original dimensions, outputting $\mathbf{X}_{\text{out}} \in \mathbb{R}^{N \times D}$. For a standard FFN, the number of parameters is $P_{\text{FFN}} = D \times rD + rD \times D = 2rD^2$, and the number of FLOPs is $F_{\text{FFN}} = 2rND^2$. In existing ViT models, the expansion ratio $r$ is typically set to 4.

**Design principles.** Recall that our goal is to develop a ViT model suitable for deployment on IoT devices with the assistance of the cloud server. In this section, we aim to explore how to reduce the complexity of the standard FFN layer while trading off the model performance to a small extent. We again consider the image classification task on the ImageNet-1K dataset for ablative studies.

TABLE 3: Ablative study on the proposed FFN. Our FFN is highlighted in bold. All results are on ImageNet-1K.

| Method | #Params | #MAdds | Top-1 Acc |
|---|---|---|---|
| Standard FFN ($r$=1) [2] | 6.0M | 0.83G | 70.8% |
| Ghost FFN ($r$=4) [47] | 6.2M | 0.87G | 71.8% |
| **PCS-FFN ($r$=4, g=4)** | **6.0M** | **0.83G** | **73.2%** |
| Standard FFN ($r$=2) [2] | 8.4M | 1.2G | 75.0% |
| Ghost FFN ($r$=2) [47] | 8.5M | 1.3G | 75.5% |
| **PCS-FFN ($r$=4, g=2)** | **8.4M** | **1.2G** | **76.1%** |
| Standard FFN ($r$=4) | 13M | 2.0G | 78.4% |

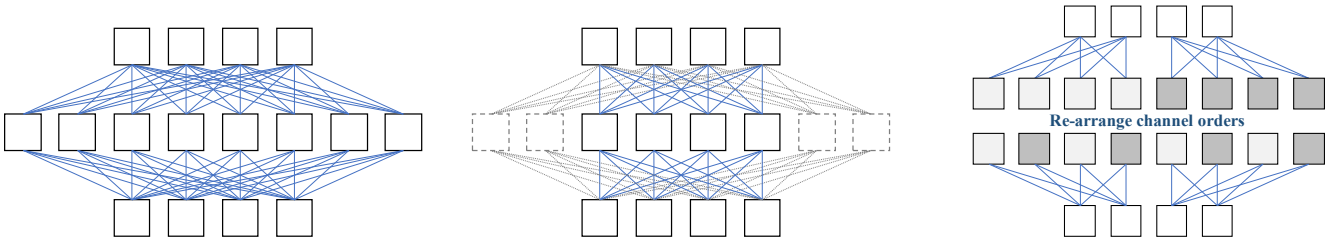
On the one hand, a straightforward (or rather naive) method for improving the efficiency of an FFN is to reduce the expansion ratio $r$ (see Fig. 5b). However, as we empirically show in Table 3, reducing $r$ results in a significant degradation in model performance, e.g., the top-1 accuracy drops by almost 8% as we reduce $r$ from 4 to 1. Despite the appealing improvement in model efficiency, such degradation in model performance renders the approach of reducing the expansion ratio $r$ unacceptable.

On the other hand, another method for improving model efficiency is to reduce the number of connections while keeping the number of channels intact. We can adjust the number of channels by adjusting the group number g. Note that when g=1, all channels of input and output are connected. When g>1, the channels are divided into g groups, the channels within each group are fully connected, and there is no connection between different groups. One realization of this idea is the groupwise connected layer (e.g., group convolution [48]). However, as pointed out by Zhang et al., a standalone groupwise connected layer is equivalent to training multiple independent small networks in parallel [14], resulting in low efficiency of utilizing channel information. Inspired by [14], we shuffle the channels between the two groupwise connected layers to achieve "cross-talk" among groups (see Fig. 5c). We name this design partially connected and shuffled FFN (PCS-FFN).

**Effectiveness.** Evidently, as suggested by the results in Table 3, PCS-FFN leads to a substantially better efficiency-performance tradeoff than existing methods. Specifically, as shown, PCS-FFN with four groups (i.e., PCS-FFN($r$=4,g=4)) is 2.4% more accurate than the standard FFN with $r$=1 (i.e., Standard FNN ($r$=1)), while being equivalent in both the number of parameters and FLOPs. As another reference for performance comparison, we also implement a Ghost FFN based on the concept (i.e., using depthwise operations to partially replace dense operations) proposed in [47]. Given the empirical evidences provided in Table 3, we set the hyperparameters of PCS-FFN to 4 for expansion ratio $r$ and 2 for number of groups $g$ for all experiments in the remainder of this paper.

## 4 Experiments

In this section, we first introduce our experimental setup, including the datasets, baselines, implementation details, and evaluation metrics used in this work. We then provide experimental comparisons with various state-of-the-art methods for image classification, object detection, and semantic segmentation tasks.




(a) Standard FFN ($r$=2) (b) Standard FFN ($r$=1) (c) Our PCS-FFN ($r$=2, g=2)

Fig. 5: (a) Standard FFN comprises two fully connected linear layers to first expand the number of channels by a ratio $r$ and then compress them back to the original number of channels. (b) One straightforward way to reduce model parameters is by reducing the expansion ratio $r$. (c) Instead, we propose using partially connected linear layers with rearranging channels between the two layers. Note that both (b) and (c) achieve 2× savings in parameters.

TABLE 4: Benchmark datasets for evaluation.

| Dataset | Task | Train Size | Valid Size | Image Size |
|---|---|---|---|---|
| ImageNet-1K [46] | Image classification | 1.28M | 50K | 224×224 |
| MS COCO [49] | Object detection | 118K | 5K | 1280×800 |
| ADE20K [50] | Semantic segmentation | 20K | 2K | 512×512 |

## 4.1 Experimental Setup

### 4.1.1 Datasets

We conduct experiments on three large-scale and challenging benchmark datasets, i.e., ImageNet-1K [46], MS COCO [49], and ADE20K [50], for image classification, object detection, and semantic segmentation tasks, respectively. See Table 4 for an overview.

**ImageNet** is one of the cornerstone datasets for quantifying progression in computer vision. ImageNet-1K is a subset of ImageNet, which consists of 1.28 million training images and 50K validation images from 1K different classes.

The **MS COCO** dataset comprises over 100K densely annotated images of diverse objects from 80 categories. We use the official *train2017* split for training and compare detection performance on the official *val2017* split.

The **ADE20K** dataset is a challenging densely annotated dataset for scene understanding. It contains 150 fine-grained semantic categories with 20K, 2K, and 3K images for training, validation, and testing, respectively.

To be fair and consistent with prior works, we train on the training set and report results on the validation set for performance comparison.

### 4.1.2 Baselines

To verify the effectiveness of our proposed TFormer, we consider a wide range of state-of-the-art baselines, as follows:

**ResNet** [42] is a family of classic convolutional neural networks that have been shown to be effective across a broad spectrum of vision tasks.

**DeiT** [3], on the one hand, is the same as the original vision transformers (ViTs) [2] from the network architecture perspective. On the other hand, DeiT proposes a data-efficient training method that enables ViTs to achieve competitive performance without pretraining on millions of images.

**PVT** [51] is a multiscale ViT as opposed to the original ViT, which is single-scaled. PVT uses a progressive shrinking pyramid to gradually reduce the spatial resolution of features, and applies spatial reduction before attention to save computations.

**Swin-Mixer-T** [5] uses local but shifted window attention to approximate full attention in ViTs. Empirically, it is shown to be as effective as full attention but with a considerable amount of savings in model complexity.

**ResMLP** [52] is a vision transformer-type architecture built completely upon multilayer perceptrons.

**PoolFormer** [53] demonstrates the importance of the skeleton structure of a vision transformer architecture, where it shows that competitive performance can be achieved with a simple average pooling operation.

### 4.1.3 Evaluation Metrics

For image classification on ImageNet-1K, we consider the standard top-1 and top-5 accuracy to compare performance. For object detection on MS COCO, we use the mean average precision (AP) computed over multiple intersection over union (IoU) values as the primary metric to compare performance. Additionally, we also report performance under a single IoU value (i.e., $AP_{50}$, $AP_{75}$) and for small, medium, and large objects (i.e., $AP_S$, $AP_M$, and $AP_L$) separately. For semantic segmentation on ADE20K, we compute the IoU for each semantic category and then compare the mean IoU (mIoU) averaged over all categories.

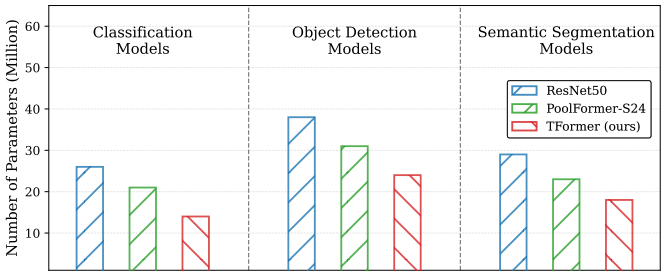



Fig. 6: Comparison of the number of parameters for different models on three vision tasks.

### 4.1.4 Implementation Details

We implement our method in Python 3.8 and PyTorch 1.8 with CUDA 11.1. All experiments are performed on

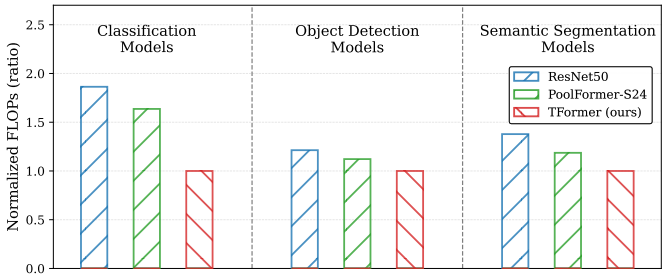



Fig. 7: Comparison of the normalized FLOPs for different models on three vision tasks.

NVIDIA 3090 GPUs. Our training setup for ImageNet-1K largely follows [3], where we use a combination of MixUp [54], CutMix [55], Cutout [56], and RandAugment [57] for data augmentation; AdamW optimizer [58] with weight decay 0.05 and an initial learning rate of 1e-3 for 300 epochs with a batch size of 1,024. For MS COCO, we use RetinaNet [59] as the detector. Following standard practice, we initialize the model with ImageNet-1K pretrained weights; we use the AdamW optimizer with an initial learning rate of 1e-4 and a batch size of 16 for 12 epochs. For ADE20K, we use the Semantic FPN [60] head and AdamW optimizer with a batch size of 32 for 40K minibatch iterations. We use an initial learning rate of 2e-4 with the polynomial decay schedule with a power of 0.9. It is worth noting that we adopt the advance training recipe introduced in [61] for training ResNet models to ensure a fair comparison. The codes will be made publicly available.

## 4.2 Experimental Results

### 4.2.1 Reduction in Model Parameter Transmission

The number of model parameters determines the quantity of data that the cloud server needs to send to the IoT device during cloud-assisted training of TFormer and the storage resources of the IoT device that the model needs to consume. Therefore, the number of model parameters is a measure of whether a model is suitable for cloud-assisted deployment on IoT devices. To this end, we compare the number of parameters that TFormer has with other models when dealing with different vision tasks, as shown in Fig. 6.

Compared to other models, our proposed TFormer has the smallest number of model parameters in all tasks. As shown in Table 5, the number of model parameters of our TFormer is only half of the number of model parameters of ResNet50. In addition, as shown in Table 6, the amount of model parameters that need to be transmitted is 57 million when using ResNet101, and 30 million when using our proposed TFormer-L. Compared with ResNet101, our proposed model saves 47% of parameters and improves AP by 2.7%. The reasons why TFormer can reduce the number of model parameters are i) replacing the multihead attention layer with the hybrid layer containing a small number of parameters and ii) introducing the PCS-FFN module, which sparsifies the connections of neural units. Therefore, TFormer has the smallest number of model parameters.

### 4.2.2 Reduction in FLOPs

Floating-point operations (FLOPs) are often used to measure model complexity and can represent the computing power requirements of the model for IoT devices. To this end, to illustrate which model is more suitable for deployment on IoT devices, we compare the FLOPs of multiple models on different tasks, as shown in Fig. 7.

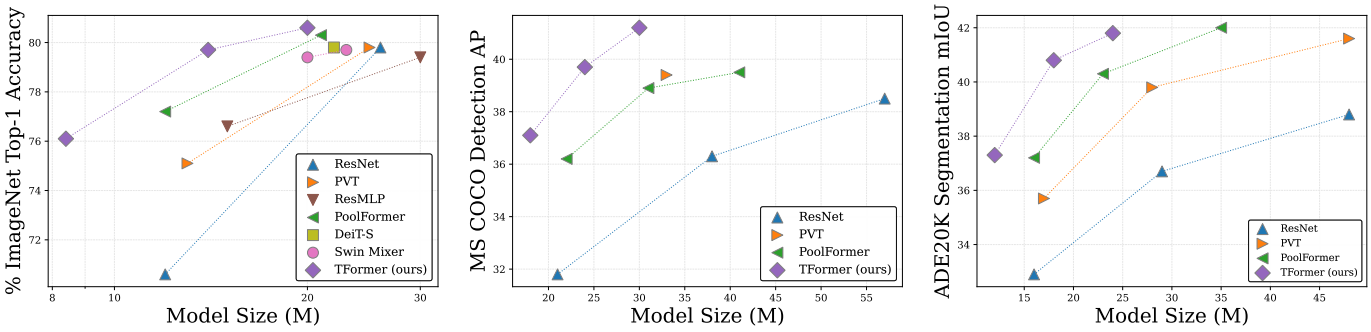
Compared to other models, our proposed TFormer has the smallest number of FLOPs in all tasks. As shown, taking our TFormer-M as the baseline, ResNet50 uses almost $2\times$ more FLOPs than our method on the image classification task; meanwhile, our method also leads to $1.6\times$ and $2.1\times$ reductions in the number of parameters and FLOPs respectively when compared to the original ViT model (i.e., DeiT-S [3]). On object detection and semantic segmentation tasks, our TFormer also has the fewest FLOPs compared to the FLOPs of ResNet50 and PoolFormer-S24. The reasons why TFormer can reduce the number of FLOPs are i) the introduction of the hybrid layer and ii) the introduction of the PCS-FFN module. Therefore, TFormer has the smallest number of FLOPs and is more suitable for deployment on resource-constrained IoT devices.

### 4.2.3 Performance Improvement

**Image Classification:** Our proposed TFormer consistently outperforms other peer models with similar or smaller numbers of parameters in the image classification task. As shown in Table 5, compared with PVT-Tiny, the top-1 accuracy of TFormer-M is 4.6% higher than that of PVT-Tiny, and the top-5 accuracy of TFormer-M is 2.4% higher than that of PVT-Tiny; In addition, as shown, TFormer outperforms models even with more parameters than it does. Compared with PVT-Tiny, the top-1 accuracy of TFormer-S is 1% higher than that of PVT-Tiny, and the top-5 accuracy of TFormer-S is 0.5% higher than that of PVT-Tiny. Compared with PVT-Small, the top-1 accuracy of TFormer-L is 0.8% higher than that of PVT-Small, and the top-5 accuracy of TFormer-L is 0.4% higher than that of PVT-Small.

**Object Detection:** Our proposed TFormer consistently outperforms other peer models with a similar or slightly larger number of parameters in the object detection task. As shown in Table 6, TFormer-L achieves 2.3 higher AP points than PoolFormer-S24 while using a similar number of parameters. TFormer-M achieves 3.5 higher AP points than PoolFormer-S12 while using a similar number of parameters. In addition, TFormer-S achieves 0.9 higher AP points than PoolFormer-S12 while using a smaller number of parameters. With only half the number of model parameters of ResNet101, the AP point of TFormer-L is 2.7 higher than that of ResNet101. In addition, we also provide a qualitative visualization between TFormer-M and the compared models in Fig. 9.

**Semantic Segmentation:** Our proposed TFormer consistently outperforms other peer models with a similar number of parameters in the semantic segmentation task. As shown in Table 7, compared with PoolFormer-S24, our TFormer-M achieves a 0.5% higher mIoU and a reduced number of 5M parameters, and our TFormer-L achieves a 1.5% higher mIoU while using a similar number of parameters. Note that compared with PoolFormer-S36, our TFormer-L reduces the




(a) ImageNet-1K classification (b) MS COCO object detection (c) ADE20K semantic segmentation

Fig. 8: Accuracy and model size on different tasks of our TFormer compared to the state-of-the-art.

TABLE 5: Image classification performance on ImageNet-1K [46]. Savings (ratio) measure the parameter or FLOPs ratios between compared approaches and our TFormers.

| Model | #Params (M) | Savings (ratio) | #MAdds (G) | Savings (ratio) | Top-1 Acc (%) | Top-5 Acc (%) |
|---|---|---|---|---|---|---|
| ResNet18 [42] | 12 | 1.4× | 1.8 | 1.5× | 70.6 | 89.6 |
| PVT-Tiny [51] | 13 | 1.5× | 1.9 | 1.6× | 75.1 | 92.4 |
| ResMLP-S12 [52] | 15 | 1.8× | 3.0 | 2.5× | 76.6 | 93.2 |
| TFormer-S (ours) | **8.4** | **1.0**× | **1.2** | **1.0**× | 76.1 | 92.9 |
| DeiT-S (0.3-12) [38] | 15 | 1.1× | 3.1 | 1.4× | 78.6 | 94.4 |
| Swin-Mixer-T/D24 [5] | 20 | 1.4× | 4.0 | 1.8× | 79.4 | 94.6 |
| ResNet50 [42] | 26 | 1.9× | 4.1 | 1.9× | 79.8 | 94.5 |
| DeiT-S [3] | 22 | 1.6× | 4.6 | 2.1× | 79.8 | 95.0 |
| PVT-Small [51] | 25 | 1.8× | 3.8 | 1.7× | 79.8 | 95.0 |
| TFormer-M (ours) | **14** | **1.0**× | **2.2** | **1.0**× | 79.7 | 94.8 |
| ResMLP-S24 | 30 | 1.5× | 6.0 | 1.9× | 79.4 | 79.4 |
| Swin-Mixer-T/D6 | 23 | 1.2× | 4.0 | 1.3× | 79.7 | 94.9 |
| PoolFormer-S24 | 21 | 1.1× | 3.6 | 1.1× | 80.3 | 95.0 |
| TFormer-L (ours) | **20** | **1.0**× | **3.2** | **1.0**× | **80.6** | **95.4** |

TABLE 6: Object detection performance on MS COCO [49].

| Model | #Params (M) | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| ResNet18 [42] | 21 | 31.8 | 49.6 | 33.6 | 16.3 | 34.3 | 43.2 |
| PoolFormer-S12 [53] | 22 | 36.2 | 56.2 | 38.2 | 20.8 | 39.1 | 48.0 |
| TFormer-S (ours) | 18 | 37.1 | 56.9 | 39.5 | 20.8 | 40.2 | 49.7 |
| ResNet50 [42] | 38 | 36.3 | 55.3 | 38.6 | 19.3 | 40.0 | 48.8 |
| PoolFormer-S24 [53] | 31 | 38.9 | 59.7 | 41.3 | 23.3 | 42.1 | 51.8 |
| TFormer-M (ours) | 24 | 39.7 | 59.9 | 42.4 | 23.8 | 43.3 | 52.9 |
| ResNet101 [42] | 57 | 38.5 | 57.8 | 41.2 | 21.4 | 42.6 | 51.1 |
| PoolFormer-S36 [53] | 41 | 39.5 | 60.5 | 41.8 | 22.5 | 42.9 | 52.4 |
| TFormer-L (ours) | 30 | 41.2 | 61.7 | 43.9 | 24.2 | 44.5 | 55.6 |

number of model parameters by 11 M at the expense of 0.2% mIoU. A qualitative comparison is provided in Fig. 10.

**Discussion.** Fig. 8 clearly shows the performance and model size of TFormer and other models on different tasks. As shown, on image classification, object detection, and semantic segmentation tasks, the proposed TFormer consistently outperforms a wide range of existing alternatives with similar or fewer parameters. The main reasons are two-fold. Firstly, it is because TFormer includes different types of pooling (i.e., max-pooling and avg-pooling) with different kernel sizes (i.e., $3 \times 3$, $5 \times 5$, $7 \times 7$, $9 \times 9$, $11 \times 11$). Different types of pooling can extract multiple types of features, and pooling of different kernel sizes can extract multiple-scale features. From [32], we observe that, multitype and multiscale features contribute to the high performance of TFormer. In addition, the introduction of the pooling operation and channel splitting greatly reduces the number of model parameters in TFormer. Secondly, it is because of the introduction of group convolution and shuffle channel techniques. The group convolution greatly reduces the number of model parameters, and the shuffle channel enables the features of different groups to flow fully. Therefore, TFormer proposed in this paper improves the performance of the model while reducing the number of model parameters, which is of great significance in the context of the Internet of Everything for applications that use the cloud server to assist IoT device deployment and update vision transformer models.

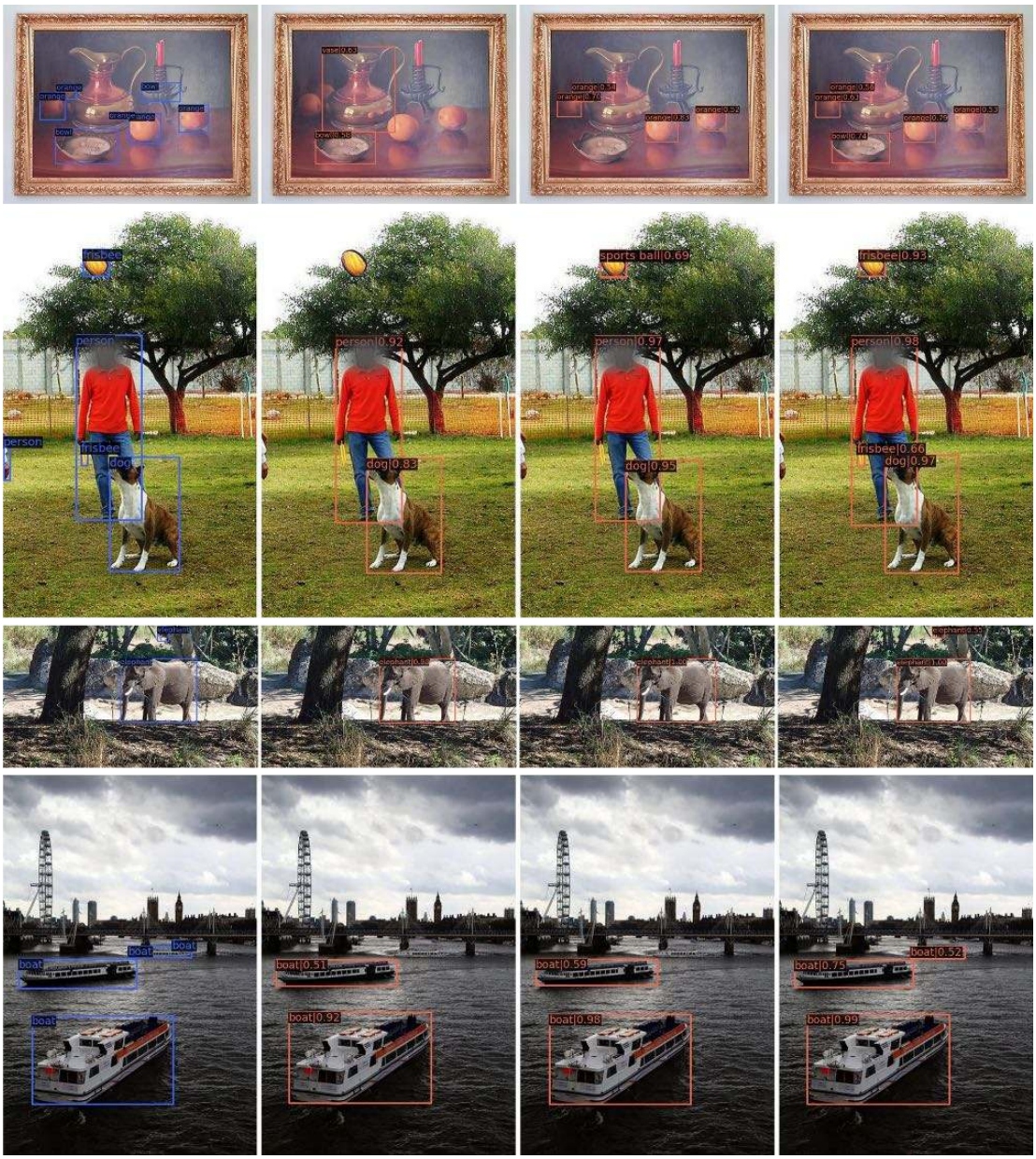



Fig. 9: Qualitative comparison on MS COCO dataset. From left to right, we show the example predictions from the ground truth, ResNet50, PoolFormer-S24, and TFormer-S24. The predicted labels with confidence scores are annotated at the top-left corners of the detection boxes.

In addition, three model variants give more options for IoT devices with different resource configurations. For example, for smart cameras with very limited computing and storage resources, TFormer-S can be preferentially configured, and for mobile phones with relatively sufficient resources, TFormer-L can be preferentially selected.

## 5 Conclusion and Future Work

This paper presented a transmission-friendly vision transformer, namely, TFormer, for IoT devices. In TFormer, a hybrid layer consisting of a nonlearnable layer and a pointwise convolution and a partially connected and shuffled feedforward network (PCS-FFN) consisting of group convolution and channel shuffle techniques is introduced to reduce the number of parameters and floating-point operations (FLOPs). In addition, the proposed hybrid layer can extract multitype and multiscale features of the data, enabling TFormer to achieve high performance. Experimental results show that TFormer can effectively improve the performance of the model on multiple tasks while reducing the number of model parameters and FLOPs.

In future work, we plan to deploy TFormer on IoT devices (e.g., Raspberry Pi 4B). Because IoT devices have the characteristics of dynamically changing available resources, to provide uninterrupted services, it is necessary to deploy multiple TFormers of different capacities. However, deploying multiple TFormers is constrained by the limited resources of the IoT device. To this end, we are going to study deploying multiple TFormers in a model parameter sharing method to save IoT device storage resources. More-

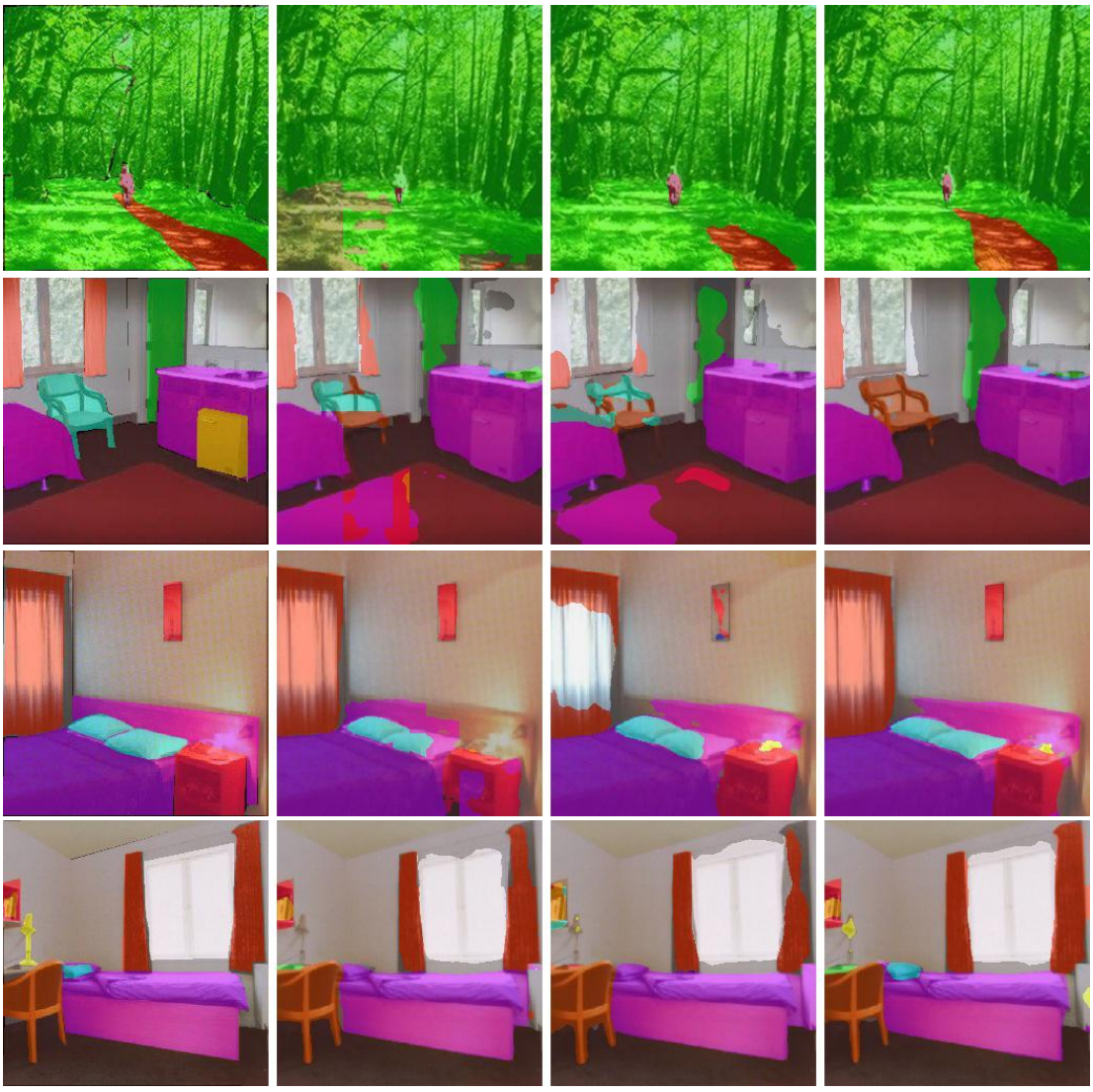


Fig. 10: Qualitative comparison on ADE20K dataset. We visualize ground truth, ResNet50, PoolFormer-S24, and TFormer-24 from left to right.

TABLE 7: Semantic segmentation performance on ADE20K [50].

| Model | #Params (M) | mIoU (%) |
|---|---|---|
| ResNet18 [42] | 16 | 32.9 |
| PVT-Tiny [51] | 17 | 35.7 |
| PoolFormer-S12 [53] | 16 | 37.2 |
| TFormer-S (ours) | 12 | 37.3 |
| ResNet50 [42] | 29 | 36.7 |
| PVT-Small [51] | 28 | 39.8 |
| PoolFormer-S24 [53] | 23 | 40.3 |
| TFormer-M (ours) | 18 | 40.8 |
| ResNet-101 [42] | 48 | 38.8 |
| PVT-Medium [51] | 48 | 41.6 |
| PoolFormer-S36 [53] | 35 | 42.0 |
| TFormer-L (ours) | 24 | 41.8 |

over, in addition to reducing the number of parameters, we will also explore ways to reduce the amount of data and better processing methods.

## ACKNOWLEDGEMENTS

C. Ding was supported by the Fundamental Research Funds for the Central Universities (2021RC272), the National Natural Science Foundation of China (62202039), and the China Postdoctoral Science Foundation (2021M700364); Z. Lu was supported by the National Natural Science Foundation of China (62106097) and the China Postdoctoral Science Foundation (2021M691424); S. Wang was supported by the National Natural Science Foundation of China (61922017).

## REFERENCES

[1] https://infohub.delltechnologies.com/l/edge-to-core-and-the-internet-of-things-2/internet-of-things-and-data-placement, [Online; accessed 9-July-2022].

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proceedings of the International Conference on Learning Representation*, 2021, pp. 1–21.

[3] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers & distillation through attention," in *Proceedings of the International Conference on Machine Learning*, 2021, pp. 10 347–10 357.

[4] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proceedings of the 16th European Conference on Computer Vision*, 2020, pp. 213–229.

[5] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9992–10 002.

[6] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. S. Torr, and L. Zhang, "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6881–6890.

[7] R. Strudel, R. G. Pinel, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7242–7252.

[8] Y. Ma, W. Liang, J. Li, X. Jia, and S. Guo, "Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 196–210, 2022.

[9] C. Ding, A. Zhou, X. Liu, X. Ma, and S. Wang, "Resource-aware feature extraction in mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 321–331, 2022.

[10] S. Wang, C. Ding, N. Zhang, X. Liu, A. Zhou, J. Cao, and X. Shen, "A cloud-guided feature extraction approach for image retrieval in mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 292–305, 2021.

[11] S. Tuli, S. llager, K. Ramamohanarao, and R. Buyya, "Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 3, pp. 940–954, 2022.

[12] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min, "Energy-efficient offloading for dnn-based smart iot systems in cloud-edge environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 683–697, 2022.

[13] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 242–253, 2021.

[14] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[15] Z. Xu, L. Zhao, W. Liang, O. F. Rana, P. Zhou, Q. Xia, W. Xu, and G. Wu, "Energy-aware inference offloading for dnn-driven applications in mobile edge clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 4, pp. 799–814, 2021.

[16] C. Ding, A. Zhou, X. Ma, and S. Wang, "Cognitive service in mobile edge computing," in *Proceedings of the IEEE International Conference on Web Services*, 2020, pp. 181–188.

[17] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proceedings of 37th IEEE International Conference on Distributed Computing Systems*, 2017, pp. 328–339.

[18] R. Han, S. Li, X. Wang, C. H. Liu, G. Xin, and L. Y. Chen, "Accelerating gossip-based deep learning in heterogeneous edge computing platforms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1591–1602, 2021.

[19] C. Blakeney, X. Li, Y. Yan, and Z. Zong, "Parallel blockwise knowledge distillation for deep neural network compression," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1765–1776, 2021.

[20] C. Ding, Z. Lu, F. Juefei-Xu, V. N. Boddeti, Y. Li, and J. Cao, "Towards transmission-friendly and robust cnn models over cloud and device," *IEEE Transactions on Mobile Computing*, pp. 1–14, 2022.

[21] J. Wu, L. Wang, Q. Pei, X. Cui, F. Liu, and T. Yang, "Hitdl: High-throughput deep learning inference at the hybrid mobile edge," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4499–4514, 2022.

[22] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "On-edge multi-task transfer learning: Model and practice with data-driven task allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1357–1371, 2020.

[23] F. Liu, P. Shu, and J. C. Lui, "Appatp: An energy conserving adaptive mobile-cloud transmission protocol," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3051–3063, 2015.

[24] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks," in *Proceedings of the Neural Information Processing Systems*, 2014, pp. 3320–3328.

[25] C. Ding, A. Zhou, Y. Liu, R. N. Chang, C.-H. Hsu, and S. Wang, "A cloud-edge collaboration framework for cognitive services," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 1489–1499, 2022.

[26] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proceedings of Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, 2017, pp. 615–629.

[27] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy, "Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints," in *Proceedings of the 14th Annual International Conference on Mobile Systems*, 2016, pp. 123–136.

[28] B. Fang, X. Zeng, and M. Zhang, "NestDNN: Resource-aware multi-tenant on-device deep learning for continuous mobile vision," in *Proceedings of 16th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 115–127.

[29] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *Proceedings of 4th International Conference on Learning Representations*, 2016, pp. 1–14.

[30] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *CoRR abs/1503.02531*, 2015, pp. 1–9.

[31] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "Spinn: Synergistic progressive inference of neural networks over device and cloud," in *Proceedings of 16th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–15.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2017, pp. 5998–6008.

[33] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "Cvt: Introducing convolutions to vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 22–31.

[34] K. Yuan, S. Guo, Z. Liu, A. Zhou, F. Yu, and W. Wu, "Incorporating convolution designs into visual transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 559–568.

[35] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z. Jiang, F. E. H. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 538–547.

[36] Y. Wang, R. Huang, S. Song, Z. Huang, and G. Huang, "Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition," *Advances in Neural Information Processing Systems*, vol. 34, pp. 11 960–11 973, 2021.

[37] L. Meng, H. Li, B.-C. Chen, S. Lan, Z. Wu, Y.-G. Jiang, and S.-N. Lim, "Adavit: Adaptive vision transformers for efficient image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 309–12 318.

[38] F. Yu, K. Huang, M. Wang, Y. Cheng, W. Chu, and L. Cui, "Width & depth pruning for vision transformers," in *AAAI Conference on Artificial Intelligence (AAAI)*, vol. 2022, 2022.

[39] Y. Tang, K. Han, Y. Wang, C. Xu, J. Guo, C. Xu, and D. Tao, "Patch slimming for efficient vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 165–12 174.

[40] M. Chen, H. Peng, J. Fu, and H. Ling, "Autoformer: Searching transformers for visual recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 270–12 280.

[41] B. Chen, P. Li, C. Li, B. Li, L. Bai, C. Lin, M. Sun, J. Yan, and W. Ouyang, "Glit: Neural architecture search for global and local image transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12–21.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[43] C. Li, Z. Yu, Y. Fu, Y. Zhang, Y. Zhao, H. You, Q. Yu, Y. Wang, C. Hao, and Y. Lin, "HW-NAS-bench: Hardware-aware neural architecture search benchmark," in *International Conference on Learning Representations*, 2021.

[44] H. Benmeziane, K. E. Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "A comprehensive survey on hardware-aware neural architecture search," *arXiv preprint arXiv:2101.09336*, 2021.
[45] N. Park and S. Kim, "How do vision transformers work?" in *Proceedings of the International Conference on Learning Representation*, 2022, pp. 1–26.
[46] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
[47] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "Ghostnet: More features from cheap operations," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1577–1586.
[48] S. Xie, R. B. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5987–5995.
[49] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the European Conference on Computer Vision*, 2014, pp. 740–755.
[50] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5122–5130.
[51] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 548–558.
[52] H. Touvron, P. Bojanowski, M. Caron, M. Cord, A. El-Nouby, E. Grave, G. Izacard, A. Joulin, G. Synnaeve, J. Verbeek *et al.*, "Resmlp: Feedforward networks for image classification with data-efficient training," *arXiv preprint arXiv:2105.03404*, 2021.
[53] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan, "Metaformer is actually what you need for vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022.
[54] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *Proceedings of the International Conference on Learning Representations*, 2018, pp. 1–13.
[55] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, and J. Choe, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6023–6031.
[56] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 13 001–13 008.
[57] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 3008–3017.
[58] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proceedings of the International Conference on Learning Representations*, 2019, pp. 1–18.
[59] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2017, pp. 2999–3007.
[60] A. Kirillov, R. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6399–6408.
[61] R. Wightman, H. Touvron, and H. Jégou, "Resnet strikes back: An improved training procedure in timm," *arXiv preprint arXiv:2110.00476*, 2021.