Transformer in Convolutional Neural Networks

Yun Liu
CVL, ETH Zurich
Switzerland

Guolei Sun
CVL, ETH Zurich
Switzerland

Yu Qiu
Nankai University
Tianjin, China

Le Zhang
UESTC
Chengdu, China

Ajad Chhatkuli
CVL, ETH Zurich
Switzerland

Luc Van Gool
CVL, ETH Zurich
Switzerland

Abstract

We tackle the low-efficiency flaw of vision transformer caused by the high computational/space complexity in Multi-Head Self-Attention (MHSA). To this end, we propose the Hierarchical MHSA (H-MHSA), whose representation is computed in a hierarchical manner. Specifically, our H-MHSA first learns feature relationships within small grids by viewing image patches as tokens. Then, small grids are merged into larger ones, within which feature relationship is learned by viewing each small grid at the preceding step as a token. This process is iterated to gradually reduce the number of tokens. The H-MHSA module is readily pluggable into any CNN architectures and amenable to training via backpropagation. We call this new backbone TransCNN, and it essentially inherits the advantages of both transformer and CNN. Experiments demonstrate that TransCNN achieves state-of-the-art accuracy for image recognition. Code and pretrained models are available at https://github.com/yun-liu/TransCNN. This technical report will keep updating by adding more experiments.

1 Introduction

In the last decade, convolutional neural networks (CNN) have been the go-to architecture in computer vision, owing to their powerful capability in learning representations from images/videos [1–11]. Meanwhile, in another field of natural language processing (NLP), the transformer architecture [12] has been the de-facto standard to handle long-range dependencies [13, 14]. Transformer relies heavily on self-attention to model global relationships of sequence data. Although global modelling is also essential for vision tasks, the 2D/3D structures of vision data make it less straightforward to apply transformers therein. This predicament was recently broken by Dosovitskiy *et al.* [15], by applying a pure transformer to sequences of image patches.

Motivated by [15], a large amount of literature on vision transformer has emerged to resolve the problems caused by the domain gap between computer vision and NLP [16–20]. From our point of view, one major problem of vision transformers is that the sequence length of image patches is much longer than that of tokens (words) in an NLP application, thus leading to high computational/space complexity when computing the Multi-Head Self-Attention (MHSA). Some efforts have been dedicated to resolving this problem. PVT [18] and MViT [20] downsample the feature to compute attention in a reduced length at the cost of losing contextual details. Swin Transformer [17] computes attention within small windows to model local relationships. It gradually enlarges the receptive field through shifting windows and stacking more layers. From this point of view, Swin Transformer [17] may still be suboptimal because it works in a similar manner to CNN and needs many layers to model long-range dependencies [15]. Instead of computing the attention score in the spatial dimension,

Preprint. Under review.

CoaT [19] computes attention in a channel-wise manner and thus may be less effective in modelling global feature dependencies [21].

We propose Hierarchical MHSA (H-MHSA) to make self-attention computation in transformer *flexible* and *efficient*. Specifically, we first split an image into patches, each of which is treated in the same way as a token [15]. Instead of computing attention across all patches, we further group patches into small grids and compute attention within each grid. This step captures local relationships and yields more discriminative local representations. Then, we merge these small grids into larger ones and compute attention within each new grid by viewing small grids at the preceding step as tokens. In this way, we essentially capture feature relationships in the larger region. This process is iterated to reduce the number of tokens gradually. Throughout this procedure, our H-MHSA computes self-attention in the increasing region sizes step by step and naturally models the global relationship in a hierarchical manner. Since each grid at each step only has a small number of tokens, we can reduce the computational/space complexity of vision transformer dramatically. We empirically observe that this strategy brings us better generalization results.

Recent efforts in transformers mainly aim at developing a unified framework for both vision and NLP tasks. Unlike those approaches, we argue that both the global dependencies and locality modelling are essential for vision tasks [22–24]. Motivated by this, we introduce a novel architecture design to inherit those merits from both transformers and CNNs, respectively. More specifically, the feature enhancement part in conventional transformers is a multilayer perceptron (MLP) for the underlying data patch. We argue this module is less powerful for "local-invariant" vision data. Combining the H-MHSA module with a more potent convolutional layer may enhance the representation ability of the network for vision data. By observing this, we exploit a new concept of **Transformer in Convolutional Neural Networks (TransCNN)**. Unlike previous transformer networks that operate on sequence data, TransCNN processes 3D feature maps directly and is thus compatible with advanced CNN techniques proposed in the last decade. TransCNN essentially inherits the merits of CNN and transformers and thus works well in learning scale/shift-invariant feature representations and modelling long-dependencies in the input data. Experiments on the benchmarking datasets demonstrate that TransCNN achieves state-of-the-art performance when compared with both CNN-based and transformer-based competitors.

2 Related Work

Convolutional neural networks. More than two decades ago, LeCun *et al.* [25] built the first deep CNN, *i.e.*, LeNet, for document recognition. About ten years ago, AlexNet [1] introduced pooling layers into CNN and pushed forward the state of the art of ImageNet classification [26] significantly. Since then, CNN has become the de-facto standard of computer vision owing to its powerful ability in representation learning. Brilliant achievements have been seen in this direction. VGGNet [2] investigates networks of increasing depth using small (3×3) convolution filters. ResNet [3] manages to build very deep networks by resolving the gradient vanishing/exploding problem with residual connections [27]. GoogLeNet [28] presents the inception architecture [29, 30] using multiple branches with different convolution kernels. ResNeXt [31] improves ResNet [3] by replacing the 3×3 convolution in the bottleneck with a grouped convolution. DenseNets [32] presents dense connections, *i.e.*, using the feature maps of all preceding layers as inputs for each layer. MobileNets [33, 34] decompose the traditional convolution into a pointwise convolution and a depthwise separable convolution for acceleration, and an inverted bottleneck is proposed for ensuring accuracy. ShuffleNets [35, 36] further decompose the pointwise convolution into pointwise group convolution and channel shuffle to reduce computational cost. MansNet [37] proposes an automated mobile neural architecture search approach to search for a model with a good trade-off between accuracy and latency. EfficientNet [38] introduces a new scaling method that uniformly scales all dimensions of depth/width/resolution of the searched architecture of MansNet [37]. The above advanced techniques are the engines driving the development of computer vision in the last decade. Instead of totally abandoning them as done in recent transformer works [16–20], we aim at introducing a generic framework that could inherit the advantages of both CNNs and transformers.

Self-attention mechanism. Inspired by the human visual system, the self-attention mechanism is usually adopted to enhance essential information and suppress noisy information. STN [39] presents the first spatial attention model through learning an appropriate spatial transformation for

each input. Chen *et al.* [40] proposed the first channel-wise attention model and achieved promising results on the image captioning task. Wang *et al.* [41] explored self-attention in deep residual networks. SENet [21] applies channel-wise attention to backbone network design and boosts the accuracy of ImageNet classification [26]. CBAM [42] sequentially applies channel-wise and spatial attention for adaptive feature refinement in deep networks. BAM [43] produces a 3D attention map by combining channel-wise and spatial attention. SK-Net [44] uses channel-wise attention to fuse multiple branches with different kernel sizes selectively. Non-local network [45] presents non-local attention for capturing long-range dependencies. ResNeSt [46] is a milestone in this direction. It applies channel-wise attention on different network branches to capture cross-feature interactions and learn diverse representations. Our work shares some similarities with these works by applying self-attention for adaptive feature refinement. The difference is that we propose H-MHSA to learn global relationships rather than a simple feature recalibration using spatial or channel-wise attention in these works.

Vision transformer. Transformer [12] entirely relies on self-attention to handle long-range dependencies of sequence data. It was first proposed for NLP tasks [13, 14]. In order to apply transformers on image data, Dosovitskiy *et al.* [15] split an image into patches and treated them as tokens. Hence, a pure transformer [12] can be adopted. Such a vision transformer (ViT) attains competitive accuracy for ImageNet classification [26]. More recently, lots of efforts have been dedicated to improving ViT. T2T-ViT [47] proposes to split an image into tokens of overlapping patches so as to represent local structure by surrounding tokens. CaiT [48] builds a deeper transformer network by introducing a per-channel weighting and specific class attention. DeepViT [49] proposes Re-attention to re-generate attention maps to increase their diversity at different layers. DeiT [50] presents a knowledge distillation strategy for improving the training of ViT [15]. Srinivas *et al.* [51] tried to add the bottleneck structure to vision transformer. Some works build pyramid transformer networks to generate multi-scale features [16–20]. PVT [18] adopts convolution operation to downsample the feature map in order to reduce the sequence length in MHSA, thus reducing the computational load. Similar to PVT [18], MViT [20] utilizes pooling to compute attention on a reduced sequence length. Swin Transformer [17] computes attention within small windows and shifts windows to gradually enlarge the receptive field. CoaT [19] computes attention in the channel dimension rather than in the traditional spatial dimension. In this paper, we introduce novel designs to reduce the computational complexity of MHSA and maintain the global relationship modelling capacity of transformers. Another salient merit of the proposed method is that the new H-MHSA module could be easily pluggable into any CNN architectures and thus making the resulting architecture inherit the advantages of both CNNs and Transformers.

3 Methodology

In this section, we first provide a brief review of vision transformer [15] in §3.1. Then, we present our H-MHSA in §3.2. Finally, we present the details of TransCNN in §3.3.

3.1 Review of Vision Transformer

Transformer [12, 15] relies heavily on MHSA to model long-range relationships. Suppose $\mathbf{X} \in \mathbb{R}^{N \times C}$ denotes the input, where N and C are the number of tokens and the feature dimension of each token, respectively. We define the query $\mathbf{Q} = \mathbf{X}\mathbf{W}^q$, the key $\mathbf{K} = \mathbf{X}\mathbf{W}^k$, and the value $\mathbf{V} = \mathbf{X}\mathbf{W}^v$, where $\mathbf{W}^q \in \mathbb{R}^{C \times C}$, $\mathbf{W}^k \in \mathbb{R}^{C \times C}$, and $\mathbf{W}^v \in \mathbb{R}^{C \times C}$ are the weight matrices of linear transformations. With a mild assumption that the input and output have the same dimension, the traditional MHSA can be calculated as

$$\mathbf{A} = \mathrm{Softmax}(\mathbf{Q}\mathbf{K}^{\mathrm{T}}/\sqrt{d})\mathbf{V}, \tag{1}$$

in which $\sqrt{d}$ means an approximate normalization, and the $\mathrm{Softmax}$ function is applied to the rows of matrix. Note that we omit the concept of multiple heads here for simplicity. In Equ. (1), the matrix product of $\mathbf{Q}\mathbf{K}^{\mathrm{T}}$ first computes the similarity between each pair of tokens. Each new token is then derived over the combination of all tokens. After the computation of MHSA, a residual connection is further added to ease the optimization, like

$$\mathbf{A}' = \mathbf{A}\mathbf{W}^p + \mathbf{X}, \tag{2}$$

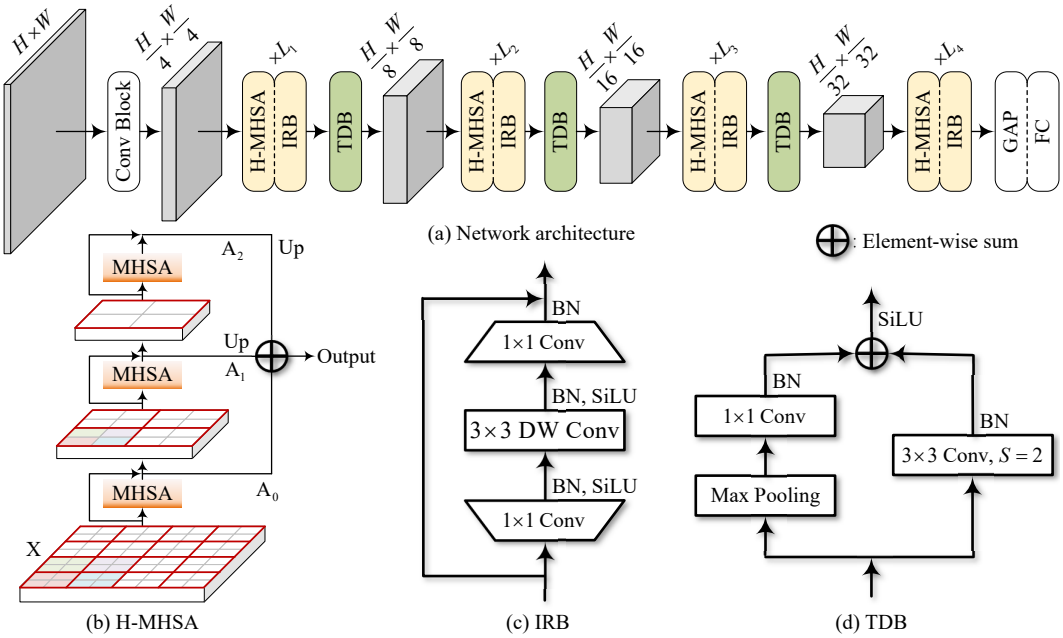

Figure 1: Illustration of the proposed TransCNN. GAP: global average pooling; FC: fully-connected layer; DW Conv: depthwise separable convolution; IRB: Inverted Residual Bottleneck [34]; TDB: Two-branch Downsampling Block. $\times L_i$ means that the H-MHSA + IRB block is repeated for L_i times. H and W denote the height and width of the input image, respectively. S denotes the stride of the convolution. SiLU [52] is a nonlinearization function.

in which $\mathbf{W}^p \in \mathbb{R}^{C\times C}$ is a weight matrix for feature projection. At last, an MLP is adopted to enhance the representation, which can be formulated as

$$\mathbf{Y} = \mathrm{MLP}(\mathbf{A}') + \mathbf{A}', \tag{3}$$

where $\mathbf{Y}$ denotes the output of a transformer block.

The computational complexity of MHSA (Equ. (1)) is

$$\Omega_{time}(\mathrm{MHSA}) = 3NC^2 + 2N^2C. \tag{4}$$

It is easy to infer that the space complexity (memory consumption) also includes the term $O(N^2)$. $O(N^2)$ could become very large for high-resolution inputs, and this limits the applicability of transformers for vision tasks. Motivated by this, we aim at reducing such complexity and maintaining the capacity of global relationship modelling without the risk of reduced performances.

3.2 Hierarchical Multi-Head Self-Attention

Here, we introduce how to reduce the computational/space complexity of Equ. (1) using our H-MHSA. Instead of computing attention across the whole input, we compute attention in a hierarchical manner so that each step only processes a limited number of tokens. Fig. 1b shows the paradigm of H-MHSA. Suppose the input feature map $\mathbf{X} \in \mathbb{R}^{H_0\times W_0\times C}$ has a height of H_0 and a width of W_0, and we have $N = H_0 \times W_0$. We divide the feature map into small grids with the size of $G_0 \times G_0$ and reshape the feature map as

$$\mathbf{X} \in \mathbb{R}^{H_0\times W_0\times C} \rightarrow \mathbf{X}' \in \mathbb{R}^{(\frac{H_0}{G_0}\times G_0)\times(\frac{W_0}{G_0}\times G_0)\times C} \rightarrow \mathbf{X}' \in \mathbb{R}^{(\frac{H_0}{G_0}\times\frac{W_0}{G_0})\times(G_0\times G_0)\times C}. \tag{5}$$

With $\mathbf{Q} = \mathbf{X}'\mathbf{W}^q$, $\mathbf{K} = \mathbf{X}'\mathbf{W}^k$, and $\mathbf{V} = \mathbf{X}'\mathbf{W}^v$, Equ. (1) is applied to generate local attention $\mathbf{A}_0$. To ease network optimization, we reshape $\mathbf{A}_0$ back to the shape of $\mathbf{X}$ through

$$\mathbf{A}_0 \in \mathbb{R}^{(\frac{H_0}{G_0}\times\frac{W_0}{G_0})\times(G_0\times G_0)\times C} \rightarrow \mathbf{A}_0 \in \mathbb{R}^{(\frac{H_0}{G_0}\times G_0)\times(\frac{W_0}{G_0}\times G_0)\times C} \rightarrow \mathbf{A}_0 \in \mathbb{R}^{H_0\times W_0\times C}, \tag{6}$$

and add a residual connection to it

$$\mathbf{A}_0 = \mathbf{X} + \mathbf{A}_0. \tag{7}$$

Since $\mathbf{A}_0$ is computed within each small $G_0 \times G_0$ grid, the computational/space complexity is reduced significantly.

For the i-th ($i > 0$) step, we view each smaller grid $G_{i-1} \times G_{i-1}$ at the (i-1)-th step as a token, which can be simply achieved by downsampling the attention feature $\mathbf{A}_{i-1}$:

$$\mathbf{A}'_{i-1} = \text{MaxPool}_{G_{i-1}}(\mathbf{A}_{i-1}) + \text{AvePool}_{G_{i-1}}(\mathbf{A}_{i-1}), \tag{8}$$

where $\text{MaxPool}_{G_{i-1}}(\cdot)$ and $\text{AvePool}_{G_{i-1}}(\cdot)$ mean to downsample $\mathbf{A}_{i-1}$ by G_{i-1} times using maximum pooling and average pooling (with kernel size and stride of G_{i-1}), respectively. Hence, we have $\mathbf{A}'_{i-1} \in \mathbb{R}^{H_i \times W_i \times C}$ with $H_i = H_0/(G_0 G_1 \cdots G_{i-1})$ and $W_i = W_0/(G_0 G_1 \cdots G_{i-1})$. Then, we divide $\mathbf{A}'_{i-1}$ into $G_i \times G_i$ grids and reshape it:

$$\mathbf{A}'_{i-1} \in \mathbb{R}^{H_i \times W_i \times C} \rightarrow \mathbf{A}'_{i-1} \in \mathbb{R}^{(\frac{H_i}{G_i} \times G_i) \times (\frac{W_i}{G_i} \times G_i) \times C} \rightarrow \mathbf{A}'_{i-1} \in \mathbb{R}^{(\frac{H_i}{G_i} \times \frac{W_i}{G_i}) \times (G_i \times G_i) \times C}. \tag{9}$$

With $\mathbf{Q} = \mathbf{A}'_{i-1}\mathbf{W}^q$, $\mathbf{K} = \mathbf{A}'_{i-1}\mathbf{W}^k$, and $\mathbf{V} = \mathbf{A}'_{i-1}\mathbf{W}^v$, Equ. (1) is called to obtain the attention feature $\mathbf{A}_i$. $\mathbf{A}_i$ is reshaped back to the shape of the input, like

$$\mathbf{A}_i \in \mathbb{R}^{(\frac{H_i}{G_i} \times \frac{W_i}{G_i}) \times (G_i \times G_i) \times C} \rightarrow \mathbf{A}_i \in \mathbb{R}^{(\frac{H_i}{G_i} \times G_i) \times (\frac{W_i}{G_i} \times G_i) \times C} \rightarrow \mathbf{A}_i \in \mathbb{R}^{H_i \times W_i \times C}, \tag{10}$$

and a residual connection is added

$$\mathbf{A}_i = \mathbf{A}'_{i-1} + \mathbf{A}_i. \tag{11}$$

This process is iterated until $H_i \times W_i$ is small enough to run Equ. (1) directly without grid splitting. The final output of H-MHSA is

$$\text{H-MHSA}(\mathbf{X}) = (\mathbf{A}_0 + \cdots + \text{Upsample}(\mathbf{A}_M))\mathbf{W}^p + \mathbf{X}, \tag{12}$$

where $\text{Upsample}(\cdot)$ means to upsample an attention feature to the original size, $\mathbf{W}^p$ has the same meaning as Equ. (2), and M is the maximum number of steps. In this way, H-MHSA can model global relationships, equivalent to traditional MHSA.

It is easy to show that, with a mild assumption that all G_i is the same, the computational complexity of H-MHSA is approximately

$$\Omega_{time}(\text{H-MHSA}) = 3NC^2 + 2NG_0^2C \tag{13}$$

Compared to Equ. (4), we reduce computational complexity significantly, *i.e.*, from $O(N^2)$ to $O(NG_0^2)$, where G_0^2 can be much smaller than N. The same conclusion can be easily derived for space complexity. Suppose we have a 1024×1024 input, PVT [18] can only downsample it into $1/8$ scale, so its computational/space complexity is approximately proportional to $N \cdot N/8^2 = 1024^2 \times 1024^2/8^2 = 16G$, while our H-MHSA only has an approximate complexity of $NG_0^2 = 1024^2 \times 32^2 = 1G$ when we set both G_0 and G_1 to 32 using two hierarchies. For Swin Transformer [17], it uses a fixed 7×7 window to scan the input and thus needs many layers to obtain a global view of the input, while our H-MHSA can model the global relationships much more efficiently. Moreover, as the downsampling operation of Equ. (8) is parameter-free, we can set G_i values flexibly for downstream vision tasks without retraining on the ImageNet dataset [26]. In contrast, PVT [18] and Swin Transformer [17] utilize fixed settings and have to be retrained on ImageNet if we want to re-parameterize the network configuration.

3.3 Transformer in Convolutional Neural Networks

Recent efforts on transformer usually aim at building pure transformer networks to provide a unified architecture for both vision and NLP tasks [16–20]. This may not be optimal for vision tasks because those architectures are not good at learning locality representations that we argue are essential for vision data [22–24]. Motivated by this, we design the H-MHSA module in a way that it could be readily pluggable into any existing CNN architecture. In this way, the resulting network could essentially inherit the merits from both transformers and CNNs.

We follow the common practice in the vision community to preserve 3D feature maps in the network backbone and use a global average pooling layer and a fully connected layer to predict image classes. This is different from existing transformers which rely on another 1D class token to make predictions

Table 1: Network configurations of TransCNN. The parameters of building blocks are shown in brackets, with the numbers of blocks stacked. For the first stage, each convolution has C channels and a stride of S. For the other four stages, each IRB uses a $K \times K$ depthwise separable convolution and an expansion ratio of E. Note that we omit the downsampling operation after t-th stage ($t = \{2, 3, 4\}$) for simplicity. "#Params" refers to the number of parameters.

Stage	Input Size	Operator	TransCNN-Small	TransCNN-Base
1	224×224	3×3 conv.	$C = 16, S = 2$ $C = 64, S = 2$	$C = 16, S = 2$ $C = 64, S = 2$
2	56×56	H-MHSA IRB	$\begin{bmatrix} C = 64 \\ K = 5 \\ E = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C = 64 \\ K = 5 \\ E = 4 \end{bmatrix} \times 3$
3	28×28	H-MHSA IRB	$\begin{bmatrix} C = 128 \\ K = 3 \\ E = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C = 128 \\ K = 3 \\ E = 4 \end{bmatrix} \times 4$
4	14×14	H-MHSA IRB	$\begin{bmatrix} C = 256 \\ K = 5 \\ E = 6 \end{bmatrix} \times 2$	$\begin{bmatrix} C = 256 \\ K = 5 \\ E = 6 \end{bmatrix} \times 8$
5	7×7	H-MHSA IRB	$\begin{bmatrix} C = 512 \\ K = 3 \\ E = 5 \end{bmatrix} \times 2$	$\begin{bmatrix} C = 512 \\ K = 3 \\ E = 6 \end{bmatrix} \times 3$
	1×1	-	Global Average Pooling, 1000-d FC, Softmax	
#Params			13.1M	26.7M

[15, 16, 18–20, 47–50, 53–55]. We also observe that previous transformer networks [15–20, 47–50] usually adopt GELU function [56] for nonlinear activation. However, GELU functions are memory-hungry during network training. We empirically found that SiLU function [52], originally coined in [56], performs on-par with GELUs and is more memory-friendly. Hence, TransCNN uses SiLU function [52] for nonlinear activation.

The overall architecture of TransCNN is illustrated in Fig. 1. At the beginning of TransCNN, unlike previous transformers that flatten image patches [15], we apply two sequential vanilla 3×3 convolutions, each of which has a stride of 2, to downsample the input image into $1/4$ scale. Then, we stack H-MHSA and convolution blocks alternatively, which can be divided into four stages with pyramid feature scales of $1/4$, $1/8$, $1/16$, and $1/32$, respectively. The convolution block we adopt is the widely-used **Inverted Residual Bottleneck** (**IRB**, Fig. 1c) with depthwise separable convolution [34]. For feature downsampling at the end of each stage, we design a simple **Two-branch Downsampling Block** (**TDB**, Fig. 1d). It consists of two branches: one branch is a vanilla 3×3 convolution with a stride of 2; the other branch is a pooling layer and a 1×1 convolution. These two branches are fused by element-wise sum to keep more contextual information in feature downsampling. Our experiments show that TDB performs better than direct downsampling.

The configuration details of TransCNN are summarized in Tab. 1. We provide two versions of TransCNN: TransCNN-Small and TransCNN-Base. TransCNN-Base has a similar number of parameters to ResNet50 [3]. Note that we only adopt the simplest parameter settings *without careful tuning* to demonstrate the *effectiveness and generality* of the proposed concepts, *i.e.*, H-MHSA and TransCNN. For example, we use the typical numbers of channels, *i.e.*, 64, 128, 256, and 512. The dimension of each head in MHSA is set to a typical value of 64. We believe that a delicate engineering tuning on those parameter settings could further boost the performance but is out of the scope of this paper.

4 Experiments

This section evaluates the proposed TransCNN for image classification on the ImageNet dataset [26]. We first provide ablation studies of TransCNN for better understanding. Then, we compare TransCNN to existing CNN- and transformer-based networks. At last, we further validate the superiority of TransCNN by applying it to object detection and instance segmentation on the popular MS-COCO dataset [57].

Table 2: Ablation studies for various design choices of this paper on the ImageNet validation set [26]. The configuration of TransCNN-Base is adopted for all experiments.

Design	3×3 IRB	Default IRB	1^{st}-level H-MHSA	2^{nd}-level H-MHSA	TDB	Top-1 Acc.
1	✔					77.0
2		✔				77.6
3		✔	✔			79.2
4		✔		✔		79.3
5		✔	✔	✔		79.9
6		✔	✔	✔	✔	**80.1**

4.1 Experimental Setup

ImageNet dataset [26] consists of 1.28M training images and 50K validation images from 1000 categories. We adopt the training set to train our network and the validation set to test the performance. We implement TransCNN using the popular PyTorch framework [58]. For a fair comparison, we follow the same training protocol as DeiT [50], which is the standard protocol for training transformer networks nowadays. Specifically, the input images are randomly cropped to 224×224 pixels, followed by random horizontal flipping and *mixup* [59] for data augmentation. Label smoothing [29] is used to avoid overfitting. The AdamW optimizer [60] is adopted with the momentum of 0.9, the weight decay of 0.05, and a mini-batch size of 128 per GPU by default. The initial learning rate is set to 8×10^{-4}, which decreases following the cosine learning rate schedule [61]. The training process lasts for 300 epochs on eight NVIDIA Tesla V100 GPUs. For model evaluation, we apply a center crop of 224×224 pixels on validation images to evaluate the recognition accuracy. We report the top-1 classification accuracy on the validation set as well as the number of parameters and the number of FLOPs for various models. Note that for ablation studies, we utilize a mini-batch size of 64 and 100 training epochs to save time. Moreover, only two hierarchies are enough for 224×224 inputs. We set $G_0 = \{8, 4, 2\}$ for t-th stage, $t = \{2, 3, 4\}$, respectively. The fifth stage can be processed directly. Besides, we adopt $\mathbf{Q} = \mathbf{A}_0\mathbf{W}^q$ rather than $\mathbf{Q} = \mathbf{A}'_0\mathbf{W}^q$ for the second hierarchy to omit an upsampling operation in Equ. (12).

4.2 Ablation Studies

In this part, we evaluate various design choices of the proposed TransCNN. As discussed above, here, we only train all ablation models for 100 epochs to save time. The batch size and learning rate are also reduced by half accordingly. The configuration of TransCNN-Base is adopted for these ablation studies.

Main components of TransCNN. We start with a pure CNN architecture by removing H-MHSA, replacing TDB with its single pooling branch, and utilizing 3×3 depthwise separable convolutions for all IRB. This baseline variant is just like MobileNetV2 [34]. From Tab. 2, we can see that the top-1 accuracy of this baseline on the ImageNet validation set [26] is 77.0%. Then, we reform this baseline with the default IRB setting of TransCNN-Base, *i.e.*, using 5×5 depthwise separable convolutions for IRB in the 2^{nd} and 4^{th} stages, inspired by [38]. The top-1 accuracy is improved from 77.0% to 77.6%. Next, we add the 1^{st} level of H-MHSA. Note that we set $G_1 = 7$ for all stages to align the setting with Swin Transformer [17]. Introducing attention into CNN significantly improves the accuracy by 1.6%. We also validate the 2^{nd} level of H-MHSA, like PVT [18]. It achieves a similar result to the 1^{st}-level H-MHSA. After that, we use the complete and default version of H-MHSA, which boosts the classification accuracy to 79.9%. H-MHSA not only improves accuracy, but also has the ability to process very large images, which is difficult for other methods [17, 18], as discussed in §3.2. At last, we add TDB into the network architecture for feature map downsampling, further improving the accuracy to 80.1%. The above experimental results suggest that all design choices of TransCNN are effective and necessary.

A pure transformer version of TransCNN *vs.* PVT [18]. When we remove all depthwise separable convolutions from TransCNN and train the resulting transformer network for 100 epochs, it achieves 77.7% top-1 accuracy on the ImageNet validation set [26]. In contrast, the well-known transformer network, PVT [18], attains 75.8% top-1 accuracy under the same condition. This suggests

Table 3: Comparison to state-of-the-art methods on the ImageNet validation set [26]. "*" indicates the performance of a method using the default training setting in the original paper. "#Params" and "#FLOPs" refer to the number of parameters and the number of FLOPs, respectively.

Architecture	Model	Input	#Params	#FLOPs	Top-1 Acc.
CNN	ResNet18* [3]	224×224	11.7M	1.8G	69.8
	ResNet18 [3]	224×224	11.7M	1.8G	68.5
Transformer	DeiT-Ti/16 [50]	224×224	5.7M	1.3G	72.2
	PVT-Tiny [18]	224×224	13.2M	1.9G	75.1
TransCNN-Small		224×224	13.1M	1.8G	79.3
CNN	ResNet50* [3]	224×224	25.6M	4.1G	76.1
	ResNet50 [3]	224×224	25.6M	4.1G	78.5
	ResNet101* [3]	224×224	44.7M	7.9G	77.4
	ResNet101 [3]	224×224	44.7M	7.9G	79.8
	ResNeXt50-32x4d* [31]	224×224	25.0M	4.3G	77.6
	ResNeXt50-32x4d [31]	224×224	25.0M	4.3G	79.5
	ResNeXt101-32x4d* [31]	224×224	44.2M	8.0G	78.8
	ResNeXt101-32x4d [31]	224×224	44.2M	8.0G	80.6
	RegNetY-4G [62]	224×224	20.6M	4.0G	80.0
	RegNetY-8G [62]	224×224	39.2M	8.0G	81.7
	ResNeSt-50 [46]	224×224	27.5M	5.4G	81.1
	ResNeSt-101 [46]	224×224	48.3M	10.3G	83.0
Transformer	T2T-ViT$_t$-14 [47]	224×224	21.5M	5.2G	80.7
	T2T-ViT$_t$-19 [47]	224×224	39.2M	8.4G	81.4
	TNT-S [53]	224×224	23.8M	5.2G	81.3
	TNT-B [53]	224×224	65.6M	14.1G	82.8
	PVT-Small [18]	224×224	24.5M	3.8G	79.8
	PVT-Medium [18]	224×224	44.2M	6.7G	81.2
	PVT-Large [18]	224×224	61.4M	9.8G	81.7
	Swin-T [17]	224×224	28.3M	4.5G	81.3
	Swin-S [17]	224×224	49.6M	8.7G	83.0
	Swin-B [17]	224×224	87.8M	15.4G	83.3
	ViT-B/16 [15]	384×384	86.6M	55.4G	77.9
	ViT-L/16 [15]	384×384	304.3M	190.7G	76.5
	DeiT-S/16 [50]	224×224	22.1M	4.6G	79.8
	DeiT-B/16 [50]	224×224	86.6M	17.6G	81.8
TransCNN-Base		224×224	26.7M	4.0G	82.2

that our proposed Hierarchical Multi-Head Self-Attention (H-MHSA) is very effective in feature representation learning. Note that H-MHSA also has the capability to handle very large images, unlike other transformer networks, as discussed in the main paper.

SiLU [52] *vs.* GELU [56]. In this paper, we adopt SiLU function [52] for nonlinearization rather than the widely-used GELU function [56] in transformers [12, 15]. Here, we evaluate the effect of this choice. The proposed TransCNN with SiLU [52] attains 80.1% top-1 accuracy on the ImageNet validation set [26] when trained for 100 epochs. TransCNN with GELU [56] gets 79.7% top-1 accuracy, slightly worse than SiLU [52]. When using a batch size of 128 per GPU, TransCNN with SiLU [52] occupies 20.2GB GPU memory in the training phase, while TransCNN with GELU [56] occupies 23.8GB GPU memory. Hence, TransCNN with SiLU [52] can achieve slightly better performance with less GPU memory consumption.

4.3 Comparison with State-of-the-art Network Architectures

After establishing the effectiveness of key modules in TransCNN, we summarize the results obtained by our method and compare it with several state-of-the-art network architectures, including CNN-based ones like ResNet [3], ResNeXt [31], RegNetY [62], ResNeSt [46], and transformer-based ones like ViT [15], DeiT [50], T2T-ViT [47], TNT [53], PVT [18], Swin Transformer [17]. Besides the top-1 accuracy on the ImageNet validation set [26], we also report the numbers of parameters

Table 4: Object detection performance on the MS-COCO `val2017` set [57]. "#Params" refers to the number of parameters.

Backbone	#Params	RetinaNet [63]					
		AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet18 [3]	21.3M	31.8	49.6	33.6	16.3	34.3	43.2
PVT-Tiny [18]	23.0M	36.7	56.9	38.9	22.6	38.8	50.0
TransCNN-Small (Ours)	22.8M	**38.8**	**59.8**	**41.3**	**23.8**	**42.6**	**50.6**
ResNet50 [3]	37.7M	36.3	55.3	38.6	19.3	40.0	48.8
PVT-Small [18]	34.2M	40.4	61.3	43.0	25.0	42.9	55.7
TransCNN-Base (Ours)	36.5M	**43.4**	**64.2**	**46.5**	**27.0**	**47.4**	**56.7**

Table 5: Instance segmentation performance on the MS-COCO `val2017` set [57]. "#Params" refers to the number of parameters. AP^b and AP^m are for bounding box AP and mask AP, respectively.

Backbone	#Params	Mask R-CNN [5]					
		AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet18 [3]	31.2M	34.0	54.0	36.7	31.2	51.0	32.7
PVT-Tiny [18]	32.9M	36.7	59.2	39.3	35.1	56.7	37.3
TransCNN-Small (Ours)	32.7M	**40.5**	**63.0**	**43.9**	**37.5**	**59.8**	**40.3**
ResNet50 [3]	44.2M	38.0	58.6	41.4	34.4	55.1	36.7
PVT-Small [18]	44.1M	40.4	62.9	43.8	37.8	60.1	40.3
TransCNN-Base (Ours)	46.4M	**44.0**	**66.4**	**48.5**	**40.2**	**63.3**	**43.2**

and FLOPs of each model. The results are summarized in Tab. 3. We can observe that TransCNN achieves state-of-the-art performance. Specifically, with similar numbers of parameters and FLOPs, TransCNN-Small outperforms its counterparts by a large margin, *i.e.*, 9.5%, 7.1%, and 4.2% higher in classification accuracy than ResNet18 [3], DeiT-Ti [50], and PVT-Tiny [18], respectively. TransCNN-Base also achieves significantly better accuracy than other counterparts with a similar number of parameters. When compared to the counterparts with much more parameters, TransCNN-Base attains very competitive accuracy. In our experiments, we also find that, when increasing the number of training epochs from 100 to 300, the accuracy of TransCNN-Base is improved from 80.1 (see in Tab. 2) to 82.2%, while PVT-Small [18] is improved from 75.8% to 79.8% under the same setting. The performance gap between TransCNN and PVT with 100 training epochs is larger than that with 300 training epochs. This may imply that the transformer and CNN composition would lead to faster network convergence over previous pure transformer networks. These experiments demonstrate the effectiveness of TransCNN in fundamental image recognition. Note that TransCNN can provide feature pyramids needed for many downstream computer vision tasks, while some transformer networks such as ViT [15], DeiT [50], T2T-ViT [47], and TNT [53], are particularly designed for image classification.

4.4 Object Detection and Instance Segmentation

Since object detection and instance segmentation are fundamental tasks in computer vision, we apply the proposed TransCNN-Base to both tasks to further evaluate the effectiveness of TransCNN. Specifically, we utilize two well-known detectors, *i.e.*, RetinaNet [63] for object detection and Mask R-CNN [5] for instance segmentation. TransCNN is compared to ResNet [3] and another popular transformed-based network, *i.e.*, PVT [18], by only replacing the backbone of the above two detectors. Experiments are conducted on the large-scale MS-COCO dataset [57] by training on `train2017` set ($\sim$118K images) and evaluating on `val2017` set (5K images). We adopt MMDection toolbox [64] for experiments and follow the experimental settings of PVT [18] for a fair comparison. During training, we initialize the backbone weights with the ImageNet-pretrained models. The detectors are fine-tuned using the AdamW optimizer [60] with an initial learning rate of 1×10^{-4} that is decreased by 10 times after the 8^{th} and 11^{th} epochs, respectively. The whole training lasts for 12 epochs with a batch size of 16. Each training image is resized to a shorter side of 800 pixels, but the longer side is not allowed to exceed 1333 pixels. During testing, each image is fixed to a shorter side of 800 pixels. We set $G_0 = \{16, 8, 4\}$ for t-th stage, $t = \{2, 3, 4\}$, respectively. The fifth stage is processed directly.

The results are displayed in Tab. 4. We can see that TransCNN substantially improves the accuracy over other network architectures in all cases with a similar number of parameters. Specifically, when RetinaNet [63] is adopted as the detector, TransCNN-Base attains 7.1%, 8.9%, 7.9% higher performance over ResNet50 [3] and 3.0%, 2.9%, 3.5% higher performance over PVT [18] in terms of AP, AP_{50}, and AP_{75}, respectively. For Mask R-CNN [5], TransCNN-Base achieves 3.9%, 3.6%, and 4.9% higher results than PVT [18] in terms of AP^b, AP^b_{50}, and AP^b_{75} (bounding box metrics), respectively. TransCNN-Base is 2.4%, 3.1%, and 3.0% better than PVT [18] in terms of AP^m, AP^m_{50}, and AP^m_{75} (mask metrics), respectively. Such significant improvement in object detection and instance segmentation shows the superiority of TransCNN in learning effective representations, making TransCNN have the potential to be applied to various computer vision tasks.

5 Conclusion

This paper tackles the low-efficiency flaw of vision transformer caused by the high computational/space complexity of MHSA. To this end, we propose a hierarchical framework for computing MHSA, *i.e.*, H-MHSA, in order to decrease the computational/space complexity. Compared to previous counterparts [18, 17] in this direction, H-MHSA has two significant advantages: i) modelling global dependencies of the input directly and ii) having the ability to process large input images with ease. Moreover, we propose to plug the flexible H-MHSA into CNNs, instead of using an MLP after attention computation for feature enhancement as in traditional vision transformers [15]. Hence, the proposed TransCNN inherits the merits of both transforms and CNNs, compatible with previous advanced transform [47–50] and CNN [28, 3, 31, 34] techniques. Experiments on image classification, object detection, and instance segmentation demonstrate the effectiveness and potential of TransCNN in representation learning.

The limitation of this paper would be that we only adopt some typical parameter settings without carefully fine-tuning so that these parameter settings may not be optimal. The reason that we only use typical settings is that we want to show the generality of TransCNN, as discussed in §3.3. In the future, we believe that neural architecture search techniques can be applied to TransCNN for optimal settings, just like CNN [38, 37]. Of course, this is out of the scope of this paper.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *NeurIPS*, pages 1097–1105, 2012. 1, 2

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE CVPR*, pages 770–778, 2016. 2, 6, 8, 9, 10

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE TPAMI*, 39(6):1137–1149, 2016.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE ICCV*, pages 2961–2969, 2017. 9, 10

[6] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE CVPR*, pages 2881–2890, 2017.

[7] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Jia-Wang Bian, Le Zhang, Xiang Bai, and Jinhui Tang. Richer convolutional features for edge detection. *IEEE TPAMI*, 41(8):1939–1946, 2019.

[8] Zenglin Shi, Le Zhang, Yun Liu, Xiaofeng Cao, Yangdong Ye, Ming-Ming Cheng, and Guoyan Zheng. Crowd counting with deep negative correlation learning. In *IEEE CVPR*, pages 5382–5390, 2018.

[9] Shi-Jie Li, Yazan AbuFarha, Yun Liu, Ming-Ming Cheng, and Juergen Gall. MS-TCN++: Multi-stage temporal convolutional network for action segmentation. *IEEE TPAMI*, 2020.

[10] Guang-Yu Nie, Ming-Ming Cheng, Yun Liu, Zhengfa Liang, Deng-Ping Fan, Yue Liu, and Yongtian Wang. Multi-level context ultra-aggregation for stereo matching. In *IEEE CVPR*, pages 3283–3291, 2019.

[11] Yun Liu, Ming-Ming Cheng, Deng-Ping Fan, Le Zhang, JiaWang Bian, and Dacheng Tao. Semantic edge detection with diverse deep supervision. *arXiv preprint arXiv:1804.02864*, 2018. 1

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 6000–6010, 2017. 1, 3, 8

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019. 1, 3

[14] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *ACL*, pages 2978–2988, 2019. 1, 3

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1, 2, 3, 6, 8, 9, 10

[16] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. *arXiv preprint arXiv:2103.16302*, 2021. 1, 2, 3, 5, 6

[17] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 1, 3, 5, 7, 8, 10

[18] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021. 1, 3, 5, 6, 7, 8, 9, 10

[19] Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-Scale conv-attentional image transformers. *arXiv preprint arXiv:2104.06399*, 2021. 2, 3

[20] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. *arXiv preprint arXiv:2104.11227*, 2021. 1, 2, 3, 5, 6

[21] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-Excitation networks. *IEEE TPAMI*, 42(8):2011–2023, 2020. 2, 3

[22] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE TIP*, 13(9):1200–1212, 2004. 2, 5

[23] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *IEEE TPAMI*, 35(6):1397–1409, 2012.

[24] M Kivanc Mihcak, Igor Kozintsev, Kannan Ramchandran, and Pierre Moulin. Low-complexity image denoising based on statistical modeling of wavelet coefficients. *IEEE SPL*, 6(12):300–303, 1999. 2, 5

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 2, 3, 5, 6, 7, 8

[27] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. 2

[28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE CVPR*, pages 1–9, 2015. 2, 10

[29] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE CVPR*, pages 2818–2826, 2016. 2, 7

[30] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, Inception-ResNet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017. 2

[31] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE CVPR*, pages 1492–1500, 2017. 2, 8, 10

[32] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE CVPR*, pages 4700–4708, 2017. 2

[33] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2

[34] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *IEEE CVPR*, pages 4510–4520, 2018. 2, 4, 6, 7, 10

[35] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *IEEE CVPR*, pages 6848–6856, 2018. 2

[36] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In *ECCV*, pages 116–131, 2018. 2

[37] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. MnasNet: Platform-aware neural architecture search for mobile. In *IEEE CVPR*, pages 2820–2828, 2019. 2, 10

[38] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *ICML*, pages 6105–6114. PMLR, 2019. 2, 7, 10

[39] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NeurIPS*, 2015. 2

[40] Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, Wei Liu, and Tat-Seng Chua. SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning. In *IEEE CVPR*, pages 5659–5667, 2017. 3

[41] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *IEEE CVPR*, pages 3156–3164, 2017. 3

[42] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional block attention module. In *ECCV*, pages 3–19, 2018. 3

[43] Jongchan Park, Sanghyun Woo, Joon-Young Lee, and In So Kweon. BAM: Bottleneck attention module. In *BMVC*, 2018. 3

[44] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *IEEE CVPR*, pages 510–519, 2019. 3

[45] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE CVPR*, pages 7794–7803, 2018. 3

[46] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. ResNeSt: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020. 3, 8

[47] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token ViT: Training vision transformers from scratch on ImageNet. *arXiv preprint arXiv:2101.11986*, 2021. 3, 6, 8, 9, 10

[48] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021. 3

[49] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Qibin Hou, and Jiashi Feng. DeepViT: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021. 3

[50] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020. 3, 6, 7, 8, 9, 10

[51] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021. 3

[52] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. 4, 6, 8

[53] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021. 6, 8, 9

[54] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. LocalViT: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*, 2021.

[55] Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. *arXiv preprint arXiv:2103.11816*, 2021. 6

[56] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016. 6, 8

[57] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, pages 740–755, 2014. 6, 9

[58] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8026–8037, 2019. 7

[59] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 7

[60] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 7, 9

[61] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 7

[62] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *IEEE CVPR*, pages 10428–10436, 2020. 8

[63] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE ICCV*, pages 2980–2988, 2017. 9, 10

[64] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. MMDetection: Open MMLab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 9