Contextual Transformer Networks for Visual Recognition

Yehao Li, Ting Yao, *Senior Member, IEEE*,
Yingwei Pan, *Member, IEEE*, and Tao Mei, *Fellow, IEEE*

Abstract—Transformer with self-attention has led to the revolutionizing of natural language processing field, and recently inspires the emergence of Transformer-style architecture design with competitive results in numerous computer vision tasks. Nevertheless, most of existing designs directly employ self-attention over a 2D feature map to obtain the attention matrix based on pairs of isolated queries and keys at each spatial location, but leave the rich contexts among neighbor keys under-exploited. In this work, we design a novel Transformer-style module, i.e., Contextual Transformer (**CoT**) block, for visual recognition. Such design fully capitalizes on the contextual information among input keys to guide the learning of dynamic attention matrix and thus strengthens the capacity of visual representation. Technically, CoT block first contextually encodes input keys via a 3×3 convolution, leading to a static contextual representation of inputs. We further concatenate the encoded keys with input queries to learn the dynamic multi-head attention matrix through two consecutive 1×1 convolutions. The learnt attention matrix is multiplied by input values to achieve the dynamic contextual representation of inputs. The fusion of the static and dynamic contextual representations are finally taken as outputs. Our CoT block is appealing in the view that it can readily replace each 3×3 convolution in ResNet architectures, yielding a Transformer-style backbone named as Contextual Transformer Networks (**CoTNet**). Through extensive experiments over a wide range of applications (e.g., image recognition, object detection, instance segmentation, and semantic segmentation), we validate the superiority of CoTNet as a stronger backbone. Source code is available at https://github.com/JDAI-CV/CoTNet.

Index Terms—Transformer, self-attention, vision transformer, image recognition

1 Introduction

Convolutional Neural Networks (CNN) [1], [2], [3], [4], [5], [6], [7] demonstrates high capability of learning discriminative visual representations, and convincingly generalizes well to a series of Computer Vision (CV) tasks, e.g., image recognition, object detection, and semantic segmentation. The de-facto recipe of CNN architecture design is based on discrete convolutional operators (e.g., 3×3 or 5×5 convolution), which effectively impose spatial locality and translation equivariance. However, the limited receptive field of convolution adversely hinders the modeling of global/long-range dependencies, and such long-range interaction subserves numerous CV tasks [8], [9]. Recently, Natural Language Processing (NLP) field has witnessed the rise of Transformer with self-attention in powerful language modeling architectures [10], [11] that triggers long-range interaction in a scalable manner. Inspired by this, there has been a steady momentum of breakthroughs [12], [13], [14], [15], [16], [17], [18] that push the limits of CV tasks by integrating CNN-based architecture with Transformer-style modules. For example, ViT [14] and DETR [13] directly process the image patches or CNN outputs using self-attention as in Transformer. [17], [18] present a stand-alone design of local self-attention module, which can completely replace the spatial convolutions in ResNet architectures. Nevertheless, previous designs mainly hinge on the independent pairwise query-key interaction for measuring attention matrix as in conventional self-attention block (Fig. 1a), thereby ignoring the rich contexts among neighbor keys.

In this work, we ask a simple question - *is there an elegant way to enhance Transformer-style architecture by exploiting the richness of context among input keys over 2D feature map*? For this purpose, we present a unique design of Transformer-style block, named Contextual Transformer (CoT), as shown in Fig. 1b. Such design unifies both context mining among keys and self-attention learning over 2D feature map in a single architecture, and thus avoids introducing additional branch for context mining. Technically, in CoT block, we first contextualize the representation of keys by performing a 3×3 convolution over all the neighbor keys within the 3×3 grid. The contextualized key feature can be treated as a *static* representation of inputs, that reflects the *static* context among local neighbors. After that, we feed the concatenation of the contextualized key feature and input query into two consecutive 1×1 convolutions, aiming to produce the attention matrix. This process naturally exploits the mutual relations among each query and all keys for self-attention learning with the guidance of the *static* context. The learnt attention matrix is further utilized to aggregate all the input values, and thus achieves the *dynamic* contextual representation of inputs to depict the *dynamic* context. We take the combination of the *static* and *dynamic* contextual representation as the

• *The authors are with JD Explore Academy, Beijing 101111, China.*
E-mail: {yehaoli.sysu, tingyao.ustc, panyw.ustc}@gmail.com, tmei@jd.com.

Manuscript received 12 Aug. 2021; revised 17 Jan. 2022; accepted 26 Mar. 2022. Date of publication 1 Apr. 2022; date of current version 6 Jan. 2023.
This work was supported by the National Key R&D Program of China under Grant 2020AAA0108600.
(Corresponding author: Ting Yao.)
Recommended for acceptance by O. Russakovsky.
Digital Object Identifier no. 10.1109/TPAMI.2022.3164083

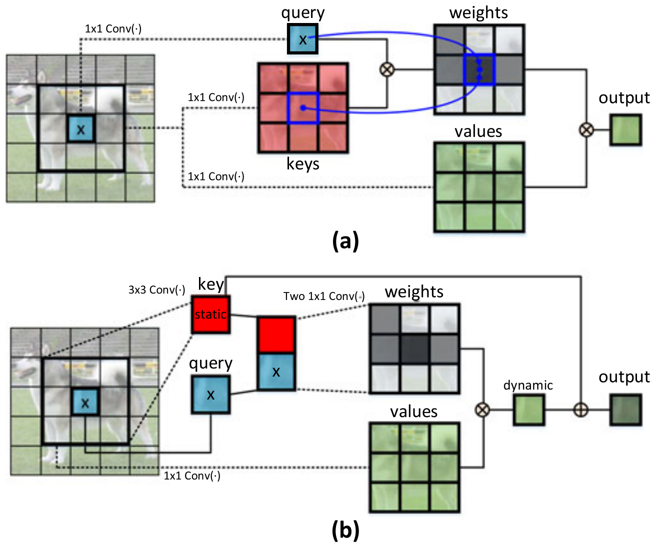


Fig. 1. Comparison between conventional self-attention and our Contextual Transformer (CoT) block. (a) Conventional self-attention solely exploits the isolated query-key pairs to measure attention matrix, but leaves rich contexts among keys under-exploited. Instead, (b) CoT block first mines the static context among keys via a 3×3 convolution. Next, based on the query and contextualized key, two consecutive 1×1 convolutions are utilized to perform self-attention, yielding the dynamic context. The static and dynamic contexts are finally fused as outputs.

final output of CoT block. In summary, our launching point is to simultaneously capture the above two kinds of spatial contexts among input keys, i.e., the *static* context via 3×3 convolution and the *dynamic* context based on contextualized self-attention, to boost visual representation learning.

Our CoT can be viewed as a unified building block, and is an alternative to standard convolutions in existing ResNet architectures without increasing the parameter and FLOP budgets. By directly replacing each 3×3 convolution in a ResNet structure with CoT block, we present a new Contextual Transformer Networks (dubbed as CoTNet) for image representation learning. Through extensive experiments over a series of CV tasks, we demonstrate that our CoTNet outperforms several state-of-the-art backbones. Notably, for image recognition on ImageNet, CoTNet obtains a 0.9% absolute reduce of the top-1 error rate against ResNeSt (101 layers). For object detection and instance segmentation on COCO, CoTNet absolutely improves ResNeSt with 1.5% and 0.86% mAP, respectively. For semantic segmentation on ADE20K, CoTNet leads to a 1.8% absolute performance improvement of mIoU against DeiT-B.

2 Related Work

2.1 Convolutional Networks

Sparked by the breakthrough performance on ImageNet dataset via AlexNet [4], Convolutional Networks (ConvNet) has become a dominant architecture in CV field. One mainstream of ConvNet design follows the primary rule in LeNet [19], i.e., stacking low-to-high convolutions in series by going deeper: 8-layer AlexNet, 16-layer VGG [5], 22-layer GoogleNet [6], and 152-layer ResNet [3]. After that, a series of innovations have been proposed for ConvNet architecture design to strengthen the capacity of visual representation. For example, inspired by split-transform-merge strategy in Inception modules, ResNeXt [20] upgrades ResNet with aggregated residual transformations in the same topology. DenseNet [21] additionally enables the cross-layer connections to boost the capacity of ConvNet. Instead of exploiting spatial dependencies in ConvNet [8], [22], SENet [23], [24] captures the interdependencies between channels to perform channel-wise feature recalibration. [25] presents a multi-scale building block for ConvNet, named Res2Net, which constructs hierarchical residual-like connections within one single residual block. HRNet [26] maintains high-resolution representations through the whole process by connecting the high-to-low resolution convolution streams in parallel and meanwhile exchanging the information across resolutions repeatedly, pursuing not only semantically strong but also spatially precise representations for position-sensitive vision problems. [7] further scales up an auto-searched ConvNet to obtain a family of EfficientNet networks, which achieve superior accuracy and efficiency. ResNeSt [27] exploits the channel-wise attention with multi-path representations in a single unified Split-Attention block, and outperforms EfficientNet with a better accuracy and latency trade-off for image recognition task.

2.2 Self-Attention in Vision

Taking the inspiration from self-attention in Transformer that continuously achieves the impressive performances in various NLP tasks, the research community starts to pay more attention to self-attention in vision scenario. The original self-attention mechanism in NLP domain [11] is devised to capture long-range dependency in sequence modeling. In vision domain, a simple migration of self-attention mechanism from NLP to CV is to directly perform self-attention over feature vectors across different spatial locations within an image. In particular, one of the early attempts of exploring self-attention in ConvNet is the non-local operation [28] that serves as an additional building block to employ self-attention over the outputs of convolutions. [12] further augments convolutional operators with global multi-head self-attention mechanism to facilitate image classification and object detection. Instead of using global self-attention over the whole feature map [12], [28] that scale poorly, [17], [18], [29] employ self-attention within local patch (e.g., 3×3 grid). Such design of local self-attention effectively limits the parameter and computation consumed by the network, and thus can fully replace convolutions across the entirety of deep architecture. Recently, by reshaping raw images into a 1D sequence, a sequence Transformer [30] is adopted to auto-regressively predict pixels for self-supervised representation learning. Next, [13], [14] directly apply a pure Transformer to the sequences of local features or image patches for object detection and image recognition. Recently, [31] designs a powerful backbone by replacing the final three 3×3 convolutions in a ResNet with global self-attention layers. Different from the conventional representation scheme adopted in ViT [14] that solely divides input image into patches, [32] first divides the inputs into several patches as “visual sentences”, and then divides them into sub-patches as “visual words”. A sub-transformer is additionally integrated into Transformer structure to excavate the features and details of smaller “visual words”. Swin Transformer [33] further upgrades ViT by constructing

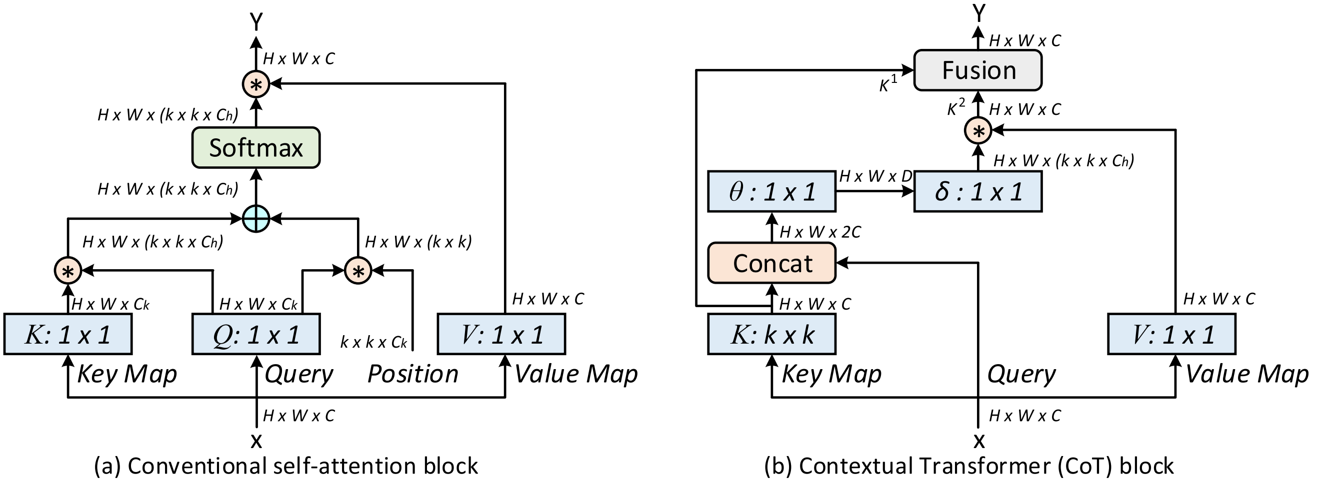


Fig. 2. The detailed structures of (a) conventional self-attention block and (b) our Contextual Transformer (CoT) block. $\oplus$ and $\circledast$ denotes the element-wise sum and local matrix multiplication, respectively.

hierarchical feature maps via merging image patches in deeper layers, which has linear computation complexity to input image size. [34] proposes twin transformers: Twins-PCPVT that explores conditional positional encodings in pyramid vision transformer and Twins-SVT that interleaves local & global attention with higher throughputs. Most recently, [35] designs a novel cross-covariance attention (XCA) which is a "transposed" version of conventional self-attention that operates across feature channels rather than tokens (words or image patches), thereby achieving linear complexity in the number of tokens.

2.3 Summary

Here we also focus on exploring self-attention for the architecture design of vision backbone. Most of existing techniques directly capitalize on the conventional self-attention and thus ignore the explicit modeling of rich contexts among neighbor keys. In contrast, our Contextual Transformer block unifies both context mining among keys and self-attention learning over feature map in a single architecture with favorable parameter budget.

3 Our Approach

In this section, we first provide a brief review of the conventional self-attention widely adopted in vision backbones. Next, a novel Transformer-style building block, named Contextual Transformer (CoT), is introduced for image representation learning. This design goes beyond conventional self-attention mechanism by additionally exploiting the contextual information among input keys to facilitate self-attention learning, and finally improves the representational properties of deep networks. After replacing 3×3 convolutions with CoT block across the whole deep architecture, two kinds of Contextual Transformer Networks, i.e., CoTNet and CoTNeXt deriving from ResNet [3] and ResNeXt [20], respectively, are further elaborated.

3.1 Multi-Head Self-Attention in Vision Backbones

Here we present a general formulation for the scalable local multi-head self-attention in vision backbones [17], [18], [29], as depicted in Fig. 2a. Formally, given an input 2D feature map X with the size of $H \times W \times C$ (H: height, W: width, C: channel number), we transform X into queries $Q = XW_q$, keys $K = XW_k$, and values $V = XW_v$ via embedding matrix (W_q, W_k, W_v), respectively. Notably, each embedding matrix is implemented as 1×1 convolution in space. After that, we obtain the local relation matrix $R \in \mathbb{R}^{H\times W\times(k\times k\times C_h)}$ between keys K and queries Q as:

$$R = K \circledast Q, \tag{1}$$

where C_h is the head number, and $\circledast$ denotes the local matrix multiplication operation that measures the pairwise relations between each query and the corresponding keys within the local $k \times k$ grid in space. Thus, each feature $R^{(i)}$ at ith spatial location of R is a $k \times k \times C_h$-dimensional vector, that consists of C_h local query-key relation maps (size: $k \times k$) for all heads. The local relation matrix R is further enriched with the position information of each $k \times k$ grid:

$$\hat{R} = R + P \circledast Q, \tag{2}$$

where $P \in \mathbb{R}^{k\times k\times C_k}$ represents the 2D relative position embeddings within each $k \times k$ grid, and is shared across all C_h heads. Next, the attention matrix A is achieved by normalizing the enhanced spatial-aware local relation matrix $\hat{R}$ with Softmax operation along channel dimension for each head: $A = \texttt{Softmax}(\hat{R})$. After reshaping the feature vector at each spatial location of A into C_h local attention matrices (size: $k \times k$), the final output feature map is calculated as the aggregation of all values within each $k \times k$ grid with the learnt local attention matrix:

$$Y = V \circledast A. \tag{3}$$

Note that the local attention matrix of each head is only utilized for aggregating evenly divided feature map of V along channel dimension, and the final output Y is the concatenation of aggregated feature maps for all heads.

3.2 Contextual Transformer Block

Conventional self-attention nicely triggers the feature interactions across different spatial locations depending on the

TABLE 1
The Detailed Structures of ResNet-50 (Left) and CoTNet-50 (Right)

stage	ResNet-50	**CoTNet-50**	output
res1	7×7 conv, 64, stride 2	7×7 conv, 64, stride 2	112×112
res2	3×3 max pool, stride 2	3×3 max pool, stride 2	56×56
	$\begin{bmatrix}1\times1, 64\\ 3\times3, 64\\ 1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\ \textbf{CoT}, 64\\ 1\times1, 256\end{bmatrix}\times3$	
res3	$\begin{bmatrix}1\times1, 128\\ 3\times3, 128\\ 1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\ \textbf{CoT}, 128\\ 1\times1, 512\end{bmatrix}\times4$	28×28
res4	$\begin{bmatrix}1\times1, 256\\ 3\times3, 256\\ 1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\ \textbf{CoT}, 256\\ 1\times1, 1024\end{bmatrix}\times6$	14×14
res5	$\begin{bmatrix}1\times1, 512\\ 3\times3, 512\\ 1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\ \textbf{CoT}, 512\\ 1\times1, 2048\end{bmatrix}\times3$	7×7
	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax	1×1
# params	$\mathbf{25.56}\times10^6$	$\mathbf{22.21}\times10^6$	
FLOPs	$\mathbf{4.12}\times10^9$	$\mathbf{3.28}\times10^9$	

The shapes and operations within a residual building block are shown inside the brackets and the number of stacked blocks in each stage is listed outside. CoTNet-50 has a slightly smaller number of parameters and FLOPs than ResNet-50.

TABLE 2
The Detailed Structures of ResNeXt-50 with a 32×4D Template (Left) and CoTNeXt-50 with a 2×48D Template (Right)

stage	ResNeXt-50 (32x4d)	**CoTNeXt-50 (2x48d)**	output
res1	7×7 conv, 64, stride 2	7×7 conv, 64, stride 2	112×112
res2	3×3 max pool, stride 2	3×3 max pool, stride 2	56×56
	$\begin{bmatrix}1\times1, 128\\ 3\times3, 128, C{=}32\\ 1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 96\\ \textbf{CoT}, 96, C{=}2\\ 1\times1, 256\end{bmatrix}\times3$	
res3	$\begin{bmatrix}1\times1, 256\\ 3\times3, 256, C{=}32\\ 1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 192\\ \textbf{CoT}, 192, C{=}2\\ 1\times1, 512\end{bmatrix}\times4$	28×28
res4	$\begin{bmatrix}1\times1, 512\\ 3\times3, 512, C{=}32\\ 1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 384\\ \textbf{CoT}, 384, C{=}2\\ 1\times1, 1024\end{bmatrix}\times6$	14×14
res5	$\begin{bmatrix}1\times1, 1024\\ 3\times3, 1024, C{=}32\\ 1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 768\\ \textbf{CoT}, 768, C{=}2\\ 1\times1, 2048\end{bmatrix}\times3$	7×7
	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax	1×1
# params	$\mathbf{25.03}\times10^6$	$\mathbf{30.05}\times10^6$	
FLOPs	$\mathbf{4.27}\times10^9$	$\mathbf{4.33}\times10^9$	

The shapes and operations within a residual building block are shown inside the brackets and the number of stacked blocks in each stage is listed outside. C denotes the number of groups within grouped convolutions. Compared to ResNeXt-50, CoTNeXt-50 has a slightly larger number of parameters but similar FLOPs.

inputs themselves. Nevertheless, in the conventional self-attention mechanism, all the pairwise query-key relations are independently learnt over isolated query-key pairs, without exploring the rich contexts in between. That severely limits the capacity of self-attention learning over 2D feature map for visual representation learning. To alleviate this issue, we construct a new Transformer-style building block, i.e., Contextual Transformer (CoT) block in Fig. 2b, that integrates both contextual information mining and self-attention learning into a unified architecture. Our launching point is to fully exploit the contextual information among neighbour keys to boost self-attention learning in an efficient manner, and strengthen the representative capacity of the output aggregated feature map.

In particular, suppose we have the same input 2D feature map $X \in \mathbb{R}^{H\times W\times C}$. The keys, queries, and values are defined as $K = X$, $Q = X$, and $V = XW_v$, respectively. Instead of encoding each key via 1×1 convolution as in typical self-attention, CoT block first employs $k\times k$ group convolution over all the neighbor keys within $k\times k$ grid spatially for contextualizing each key representation. The learnt contextualized keys $K^1 \in \mathbb{R}^{H\times W\times C}$ naturally reflect the static contextual information among local neighbor keys, and we take K^1 as the static context representation of input X. After that, conditioned on the concatenation of contextualized keys K^1 and queries Q, the attention matrix is achieved through two consecutive 1×1 convolutions (W_θ with ReLU activation function and W_δ without activation function):

$$A = [K^1, Q]W_\theta W_\delta. \tag{4}$$

In other words, for each head, the local attention matrix at each spatial location of A is learnt based on the query feature and the contextualized key feature, rather than the isolated query-key pairs. Such way enhances self-attention learning with the additional guidance of the mined static context K^1. Next, depending on the contextualized attention matrix A, we calculate the attended feature map K^2 by aggregating all values V as in typical self-attention:

$$K^2 = V \circledast A. \tag{5}$$

In view that the attended feature map K^2 captures the dynamic feature interactions among inputs, we name K^2 as the dynamic contextual representation of inputs. The final output of our CoT block (Y) is thus measured as the fusion of the static context K^1 and dynamic context K^2 through attention mechanism [36]. In particular, we first directly fuse the two contexts and obtain channel-wise global features via global average pooling, which further guide the soft attention across channels to adaptively aggregate the two contexts as the final outputs.

3.3 Contextual Transformer Networks

The design of our CoT is a unified self-attention building block, and acts as an alternative to standard convolutions in ConvNet. As a result, it is feasible to replace convolutions with their CoT counterparts for strengthening vision backbones with contextualized self-attention. Here we present how to integrate CoT blocks into existing state-of-the-art ResNet architectures (e.g., ResNet [3] and ResNeXt [20]) without increasing parameter budget significantly. Table 1 and Table 2 shows two different constructions of our Contextual Transformer Networks (CoTNet) based on the ResNet-50/ResNeXt-50 backbone, called CoTNet-50 and CoTNeXt-50, respectively. Please note that our CoTNet is flexible to generalize to deeper networks (e.g., ResNet-101).

CoTNet-50. Specifically, CoTNet-50 is built by directly replacing all the 3×3 convolutions (in the stages of res2, res3, res4, and res5) in ResNet-50 with CoT blocks. As our CoT blocks are computationally similar with the typical convolutions, CoTNet-50 has similar (even slightly smaller) parameter number and FLOPs with ResNet-50.

CoTNeXt-50. Similarly, for the construction of CoTNeXt-50, we first replace all the 3×3 convolution kernels in group convolutions of ResNeXt-50 with CoT blocks. Compared to typical convolutions, the depth of the kernels within group convolutions is significantly decreased when the number of groups (i.e., C in Table 2) is increased. In ResNeXt-50, the computational cost of group convolutions is thus reduced by a factor of C. Therefore, in order to achieve the similar parameter number and FLOPs with ResNeXt-50, we additionally reduce the scale of input feature map of CoTNeXt-50 from 32×4d to 2×48d. Finally, CoTNeXt-50 requires only 1.2× more parameters and 1.01× more FLOPs than ResNeXt-50.

3.4 Connections With Previous Vision Backbones

In this section, we discuss the detailed relations and differences between our Contextual Transformer and the previous most related vision backbones.

Blueprint Separable Convolution [37] approximates the conventional convolution with a 1×1 pointwise convolution plus a $k \times k$ depthwise convolution, aiming to reduce the redundancies along depth axis. In general, such design has some commonalities with the transformer-style block (e.g., the typical self-attention and our CoT block). This is due to that the transformer-style block also utilizes 1×1 pointwise convolution to transform the inputs into values, and the followed aggregation computation with $k \times k$ local attention matrix is performed in a similar depthwise manner. Besides, for each head, the aggregation computation in transformer-style block adopts channel sharing strategy for efficient implementation without any significant accuracy drop. Here the utilized channel sharing strategy can also be interpreted as the tied block convolution [38], which shares the same filters over equal blocks of channels.

Dynamic Region-Aware Convolution [39] introduces a filter generator module (consisting of two consecutive 1×1) to learn specialized filters for region features at different spatial locations. It therefore shares a similar spirit with the attention matrix generator in our CoT block that achieves dynamic local attention matrix for each spatial location. Nevertheless, the filter generator module in [39] produces the specialized filters based on the primary input feature map. In contrast, our attention matrix generator fully exploits the complex feature interactions between contextualized keys and queries for self-attention learning.

Bottleneck Transformer [31] is the contemporary work, which also aims to augment ConvNet with self-attention mechanism by replacing 3×3 convolution with Transformer-style module. Specifically, it adopts global multi-head self-attention layers, which are computationally more expensive than local self-attention in our CoT block. Therefore, with regard to the same ResNet backbone, BoT50 in [31] only replaces the final three 3×3 convolutions with Bottleneck Transformer blocks, while our CoT block can completely replace 3×3 convolutions across the whole deep architecture. In addition, our CoT block goes beyond typical local self-attention in [17], [18], [29] by exploiting the rich contexts among input keys to strengthen self-attention learning.

4 Experiments

In this section, we verify and analyze the effectiveness of our Contextual Transformer Networks (CoTNet) as a backbone via empirical evaluations over multiple mainstream CV applications, ranging from image recognition, object detection, instance segmentation, to semantic segmentation. Specifically, we first undertake experiments for image recognition task on ImageNet benchmark [40] by training our CoTNet from scratch. Next, after pre-training CoTNet on ImageNet, we further evaluate the generalization capability of the pre-trained CoTNet when transferred to downstream tasks of object detection & instance segmentation on COCO dataset [41] and semantic segmentation on ADE20K dataset [42].

4.1 Image Recognition

4.1.1 Setup

We conduct image recognition task on the ImageNet dataset, which consists of 1.28 million training images and 50,000 validation images derived from 1,000 classes. Both of the top-1 and top-5 accuracies on the validation set are reported for evaluation. For this task, we adopt two different training setups in the experiments, i.e., the default training setup and advanced training setup.

The default training setup is the widely adopted setting in classic vision backbones (e.g., ResNet [3], ResNeXt [20], and SENet [23]), that trains networks for around 100 epochs with standard preprocessing. Specifically, each input image is cropped into 224×224, and only the standard data augmentation (i.e., random crops and horizontal flip with 50% probability) is performed. All the hyperparameters are set as in official implementations without any additional tuning. Similarly, our CoTNet is trained in an end-to-end manner, through backpropagation using SGD with momentum 0.9 and label smoothing 0.1. We set the batch size as $B = 512$ that enables applicable implementations on an 8-GPU machine. For the first five epochs, the learning rate is scaled linearly from 0 to $\frac{0.1 \cdot B}{256}$, which is further decayed via cosine schedule [43]. As in [44], we adopt exponential moving average with weight 0.9999 during training.

For fair comparison with state-of-the-art backbones (e.g., ResNeSt [27], EfficientNet [7] and LambdaNetworks [44]), we additionally involve the advanced training setup with longer training epochs and improved data augmentation & regularization. In this setup, we train our CoTNet with 350 epochs, coupled with the additional data augmentation of RandAugment [45] and mixup [46], and the regularization of dropout [47] and DropConnect [48].

4.1.2 Performance Comparison

We compare with several state-of-the-art vision backbones with two different training settings (i.e., default and advanced training setups) on ImageNet dataset. The performance comparisons are summarized in Tables 3 and 4 for each kind of training setup, respectively. Note that we construct several variants of our CoTNet and CoTNeXt with two kinds of depthes (i.e., 50-layer and 101-layer), yielding CoTNet-50/101 and CoTNeXt-50/101. In advanced training setup, as in LambdaResNet [44], we additionally include an upgraded version of our CoTNet, i.e., SE-CoTNetD-101,

TABLE 3
Performance Comparisons with the State-of-the-Art Vision Backbones for Image Recognition on ImageNet Dataset (Default Training Setup)

Backbone	Res.	Params	GFLOPs	Top-1 Acc.	Top-5 Acc.
ResNet-50 [3]	224	25.5M	4.1	77.3	93.6
Res2Net-50 [25]	224	25.7M	4.3	78.0	93.9
ResNeXt-50 [20]	224	25.0M	4.2	78.2	93.9
SE-ResNeXt-50 [23]	224	27.6M	4.3	78.6	94.2
LR-Net-50 [29]	224	23.3M	4.3	77.3	93.6
Stand-Alone* [17]	224	18.0M	3.6	77.6	-
AA-ResNet-50 [12]	224	25.8M	4.2	77.7	93.8
BoTNet-S1-50 [31]	224	20.8M	4.3	77.7	93.7
ViT-B/16 [14]	384	-	-	77.9	-
SAN19 [18]	224	20.5M	3.3	78.2	93.9
LambdaResNet-50* [44]	224	15.0M	-	78.4	-
CoTNet-50	224	22.2M	3.3	**79.2**	**94.5**
CoTNet-50*	224	22.2M	3.3	**79.8**	**94.9**
CoTNeXt-50	224	30.1M	4.3	**79.5**	**94.5**
CoTNeXt-50*	224	30.1M	4.3	**80.2**	**95.1**
SE-CoTNetD-50	224	23.1M	4.1	**79.8**	**94.7**
SE-CoTNetD-50*	224	23.1M	4.1	**80.5**	**95.2**
ResNet-101 [3]	224	44.6M	7.9	78.5	94.2
ResNeXt-101 [20]	224	44.2M	8.0	79.1	94.4
Res2Net-101 [25]	224	45.2M	8.1	79.2	94.4
SE-ResNeXt-101 [23]	224	49.0M	8.0	79.4	94.6
LR-Net-101 [29]	224	42.0M	8.0	78.5	94.3
AA-ResNet-101 [12]	224	45.4M	8.1	78.7	94.4
CoTNet-101	224	38.3M	6.1	**80.0**	**94.9**
CoTNet-101*	224	38.3M	6.1	**80.9**	**95.3**
CoTNeXt-101	224	53.4M	8.2	**80.3**	**95.0**
CoTNeXt-101*	224	53.4M	8.2	**81.3**	**95.6**
SE-CoTNetD-101	224	40.9M	8.5	**80.5**	**95.1**
SE-CoTNetD-101*	224	40.9M	8.5	**81.4**	**95.6**

*Models with same depth (50-layer/101-layer) are grouped for efficiency comparison. * indicates the use of exponential moving average during training.*

TABLE 4
Performance Comparisons with the State-of-the-Art Vision Backbones for Image Recognition on ImageNet Dataset (Advanced Training Setup)

Backbone	Res.	Params	GFLOPs	Top-1 Acc.	Top-5 Acc.
ResNet-50 [3]	224	25.5M	4.1	78.3	94.3
CoaT-Lite Mini [49]	224	11M	2.0	78.9	-
EfficientNet-B1 [7]	240	7.8M	0.7	79.1	94.4
SE-ResNet-50 [23]	224	28.1M	4.1	79.4	94.6
XCiT-T24 [35]	224	12.1M	2.3	79.4	-
EfficientNet-B2 [7]	260	9.2M	1.0	80.1	94.9
BoTNet-S1-50 [31]	224	20.8M	4.3	80.4	95.0
ResNeSt-50-fast [27]	224	27.5M	4.3	80.6	-
ResNeSt-50 [27]	224	27.5M	5.4	81.1	-
Twins-PCPVT-S [34]	224	24.1M	3.7	81.2	-
Swin-T [33]	224	28.3M	4.5	81.3	-
CoTNet-50	224	22.2M	3.3	**81.3**	**95.6**
CoTNeXt-50	224	30.1M	4.3	**82.1**	**95.9**
SE-CoTNetD-50	224	23.1M	4.1	**81.6**	**95.8**
ResNet-101 [3]	224	44.6M	7.9	80.0	95.0
ResNet-152 [3]	224	60.2M	11.6	81.3	95.5
SE-ResNet-101 [23]	224	49.3M	7.9	81.4	95.7
TNT-S [32]	224	23.8M	5.2	81.5	95.7
EfficientNet-B3 [7]	300	12.0M	1.8	81.6	95.7
BoTNet-S1-59 [31]	224	33.5M	7.3	81.7	95.8
CoaT-Lite Small [49]	224	19.8M	4.0	81.9	-
ResNeSt-101-fast [27]	224	48.2M	8.1	82.0	-
ResNeSt-101 [27]	224	48.3M	10.2	82.3	-
LambdaResNet-101[44]	224	36.9M	-	82.3	-
XCiT-S24 [35]	224	47.6M	9.1	82.6	-
CaiT-S-24 [50]	224	46.9M	9.4	82.7	-
Twins-PCPVT-B [34]	224	56.0M	8.3	82.7	-
CoTNet-101	224	38.3M	6.1	**82.8**	**96.2**
CoTNeXt-101	224	53.4M	8.2	**83.2**	**96.4**
SE-CoTNetD-101	224	40.9M	8.5	**83.2**	**96.5**
SE-ResNet-152 [23]	224	66.8M	11.6	82.2	95.9
ConViT-B [51]	224	86.5M	16.8	82.4	95.9
BoTNet-S1-110 [31]	224	54.7M	10.9	82.8	96.3
TNT-B [32]	224	65.6M	14.1	82.9	96.3
XCiT-L24 [35]	224	189.1M	36.1	82.9	-
EfficientNet-B4 [7]	380	19.0M	4.2	82.9	96.4
CaiT-S-36 [50]	224	68.2M	13.9	83.3	-
Twins-PCPVT-L [34]	224	99.2M	14.8	83.3	-
Swin-B [33]	224	87.7M	15.4	83.3	-
BoTNet-S1-128 [31]	256	75.1M	19.3	83.5	96.5
EfficientNet-B5 [7]	456	30.0M	9.9	83.6	96.7
SE-CoTNetD-152	224	55.8M	17.0	**84.0**	**97.0**
SENet-350 [23]	384	115.2M	52.9	83.8	96.6
EfficientNet-B6 [7]	528	43.0M	19.0	84.0	96.8
BoTNet-S1-128 [31]	320	75.1M	30.9	84.2	96.9
Swin-B [33]	384	87.7M	47.0	84.2	-
EfficientNet-B7 [7]	600	66.0M	37.0	84.3	97.0
SE-CoTNetD-152	320	55.8M	26.5	**84.6**	**97.1**

Models with similar top-1/top-5 accuracy are grouped for efficiency comparison.

where the 3×3 convolutions in the res4 and res5 stages are replaced with CoT blocks under SE-ResNetD-50 [52], [53] backbone. Moreover, in default training setup, we also report the performances of our models with the use of exponential moving average for fair comparison against LambdaResNet.

As shown in Table 3, under the same depth (50-layer or 101-layer), the results across both top-1 and top-5 accuracy consistently indicate that our CoTNet-50/101 and CoTNeXt-50/101 obtain better performances against existing vision backbones with favorable parameter budget, including both ConvNets (e.g., ResNet-50/101 and ResNeXt-50/101) and attention-based models (e.g., Stand-Alone and AA-ResNet-50/101). The results generally highlight the key advantage of exploiting contextual information among keys in self-attention learning for visual recognition task. Specifically, under the same 50-layer backbones, by exploiting local self-attention in the deep architecture, LR-Net-50 and Stand-Alone exhibit better performance than ResNet-50, which ignores long-range feature interactions. Next, AA-ResNet-50 and LambdaResNet-50 enable the exploration of global self-attention over the whole feature map, and thereby boost up the performances. However, the performances of AA-ResNet-50 and LambdaResNet-50 are still lower than the stronger ConvNet (SE-ResNeXt-50) that strengthens the capacity of visual representation with channel-wise feature re-calibration. Furthermore, by fully replacing 3×3 convolutions with CoT blocks across the entirety of deep architecture in ResNet-50/ResNeXt-50, CoTNet-50 and CoTNeXt-50 outperform SE-ResNeXt-50. This confirms that unifying both context mining among keys and self-attention learning into a single architecture is an effective way to enhance representation learning and thus boost visual recognition. When additionally using exponential moving average as in LambdaResNet, the top-1 accuracy of CoTNeXt-50/101 will be further improved to 80.2% and 81.3% respectively, which is to-date the best published performance on ImageNet in default training setup.

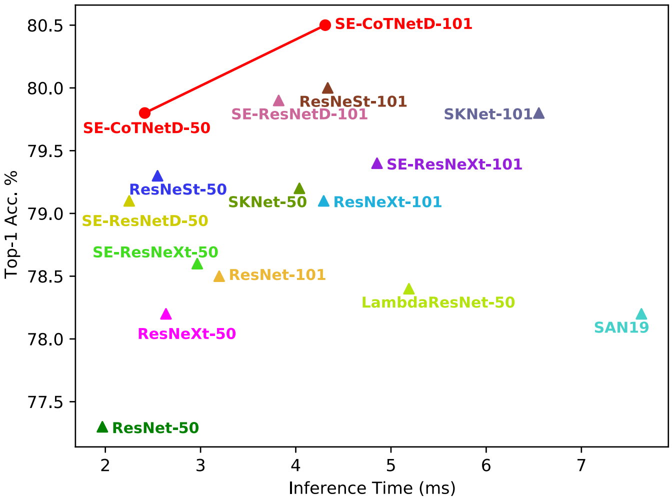


Fig. 3. Inference Time versus Accuracy Curve on ImageNet dataset (default training setup).

Similar observations are also attained in advanced training setup, as summarized in Table 4. Note that here we group all the baselines with similar top-1/top-5 accuracy or network depth. In general, our CoTNet-50 & CoTNeXt-50 or CoTNet-101 & CoTNeXt-101 perform consistently better than other vision backbones across both metrics for each group. In particular, the top-1 accuracy of our CoTNeXt-50 and CoTNeXt-101 can achieve 82.1% and 83.2%, making the absolute improvement over the best competitor ResNeSt-50 or ResNeSt-101/LambdaResNet-10 by 1.0% and 0.9%, respectively. More specifically, the attention-based backbones (BoTNet-S1-50 and BoTNet-S1-59) exhibit better performances than ResNet-50 and ResNet-101, by replacing the final three 3×3 convolutions in ResNet with global self-attention layers. LambdaResNet-101 further boosts up the performances by leveraging the computationally efficient global self-attention layers (i.e., Lambda layer) to replace the convolutional layers. Nevertheless, LambdaResNet-101 is inferior to CoTNeXt-101 which capitalizes on the contextual information among input keys to guide self-attention learning. Even under the heavy setting with deeper networks, our SE-CoTNetD-152 (320) still manages to outperform the superior backbones of BoTNet-S1-128 (320) and EfficientNet-B7 (600), sharing the similar (even smaller) FLOPs with BoTNet-S1-128 (320).

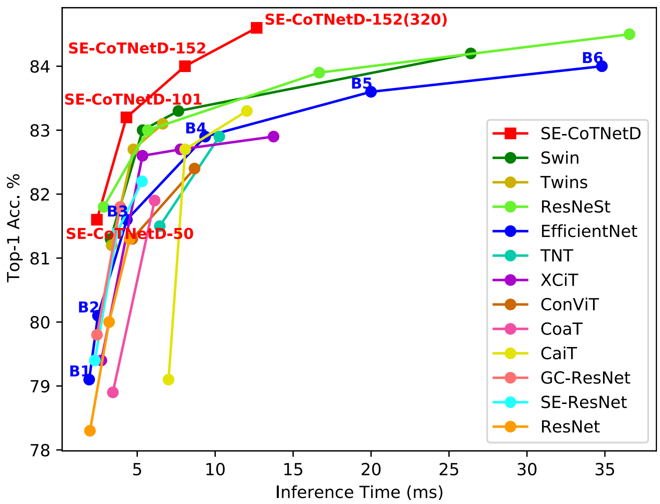


Fig. 4. Inference Time versus Accuracy Curve on ImageNet dataset (advanced training setup).

TABLE 5
Performance Comparisons across Different Ways on the Exploration of Contextual Information, I.E., Using Only Static Context (**Static Context**), Using Only Dynamic Context (**Dynamic Context**), Linearly Fusing Static and Dynamic Contexts (**Linear Fusion**), Directly Concatenating the Static and Dynamic Contexts (**Concatenate**) and the Full Version of CoT Block

	Params	GFLOPs	Top-1 Acc.	Top-5 Acc.
LR-Net-50 [29]	23.3M	4.3	77.3	93.6
BoTNet-S1-50 [31]	20.8M	4.3	77.7	93.7
AA-ResNet-50 [12]	25.8M	4.2	77.7	93.8
Static Context (CoT)	17.1M	2.7	77.1	93.5
Dynamic Context (CoT)	20.3M	3.3	78.5	94.1
Linear Fusion (CoT)	20.3M	3.3	78.7	94.2
Concatenate (CoT)	22.8M	3.7	78.9	94.3
CoT	22.2M	3.3	79.2	94.5

The backbone is CoTNet-50 and we adopt the default setup for training on ImageNet.

4.1.3 Inference Time versus Accuracy

Here we evaluate our CoTNet models with regard to both inference time and top-1 accuracy for image recognition task. Figs. 3 and 4 show the inference time-accuracy curve under both default and advanced training setups for our CoTNet and the state-of-the-art vision backbones. As shown in the two figures, we can see that our CoTNet models consistently obtain better top-1 accuracy with less inference time than other vision backbones across both training setups. In a word, our CoTNet models seek better inference time-accuracy trade-offs than existing vision backbones (e.g., SE-ResNet [23] and GC-ResNet [54]). More remarkably, compared to the high-quality backbone of EfficientNet-B6, our SE-CoTNetD-152 (320) achieves 0.6% higher top-1 accuracy, while runs 2.75× faster at inference.

4.1.4 Ablation Study

In this section, we investigate how each design in our CoT block influences the overall performance of CoTNet-50. In CoT block, we first mine the static context among keys via a 3×3 convolution. Conditioned on the concatenation of query and contextualized key, we can also obtain the dynamic context via self-attention. CoT block dynamically fuses the static and dynamic contexts as the final outputs. Here we include two variants of CoT block by directly summating or concatenating the two kinds of contexts, named as Linear Fusion and Concatenate, respectively.

Table 5 details the performances across different ways on the exploration of contextual information in CoTNet-50 backbone. Solely using static context (Static Context) for image recognition achieves 77.1% top-1 accuracy, which can be interpreted as one kind of ConvNet without self-attention. Next, by directly exploiting the dynamic context via self-attention, Dynamic Context exhibits better performance. Note that here we additionally include several runs of

TABLE 6
Performance Comparisons across Different Kernel Sizes of the Key's Convolution and Different Variants of CoT Block

Kernel Size	Params	GFLOPs	Top-1 Acc.	Top-5 Acc.
1×1	19.7M	2.9	78.6	94.1
3×3	22.2M	3.3	79.2	94.5
5×5	28.5M	4.3	79.3	94.5
3×3*	29.5M	4.5	79.4	94.5
G-CoTNet-50	28.3M	3.6	79.3	94.5
Remove Query	21.6M	3.2	79.0	94.4
Remove Key	21.6M	3.2	78.8	94.3

* *indicates the kernel sizes of convolution for query, key and value are set as 3×3.* ***G-CoTNet-50*** *is constructed by performing our contextual self-attention learning over the output global feature map of the last stage res5 in ResNet-50.* ***Remove Query*** *and* ***Remove Key*** *denotes the variant of CoT block by directly removing query or contextualized key for measuring dynamic context.*

conventional self-attention in the backbone of ResNet-50 (i.e., LR-Net-50 [29], BoTNet-S1-50 [31], and AA-ResNet-50 [12]) on ImageNet (default setup). Our Dynamic Context still manages to outperform these conventional self-attention runs. This clearly show the effectiveness of capitalizing on the contextual information among input keys to guide self-attention learning in CoTNet-50. The linear fusion of static and dynamic contexts leads to a boost of 78.7%, which basically validates the complementarity of the two contexts. Concatenate achieves comparable performances against Linear Fusion. CoT block is further benefited from the dynamic fusion via attention, and the top-1 accuracy of CoT finally reaches 79.2%.

4.1.5 Effect of Kernel Sizes of Key's Convolution

To clarify the effect of the kernel size of key's convolution in our CoT block, we compare the performances of CoTNet-50 by varying the kernel size of key's convolution in the range of {1×1, 3×3, 5×5}. As shown in the first three rows of Table 6, increasing the kernel size of key's convolution in CoT block can generally lead to performance boost. However, the parameter number & FLOPs are significantly increased. Therefore, in our experiments, we empirically set the kernel size of key's convolution in CoT block as 3×3, which seeks a better tradeoff between performance and parameter budget. In addition, we include a variant of CoT block (i.e., the fourth row of Table 6) by applying 3×3 convolution for transforming query, key and value. The top-1 score of this run is increased from 79.2% to 79.4%, while the FLOPs is increased by 35.9%. That's why we only apply 3×3 convolution for transforming key in our CoT block, as this design has a better performance-FLOPs tradeoff.

Moreover, we construct another variant of CoTNet-50 (named G-CoTNet-50) by performing contextual global self-attention learning over the output global feature map of the last stage res5 in ResNet-50. In particular, we first employ 3 stacked 5×5 convolutions over the global feature map to obtain the contextualized key (i.e., static context). Next, based on the query and contextualized key, two consecutive 1×1 convolutions are leveraged to perform global self-attention, yielding the dynamic context. Both of static and dynamic contexts are finally fused as outputs. Finally, in comparison to CoTNet-50 (3×3 convolution), G-CoTNet-50 achieves similar top-1 accuracy score (79.3%), while the GFLOPs is increased by 9.1%.

Recall that in our CoT block, we concatenate query and contextualized key to obtain dynamic context via self-attention. In order to validate such design, we also include two variants of CoT block by directly removing query or contextualized key for measuring dynamic context, named as Remove Query and Remove Key, respectively. As shown in Table 6, by additionally exploiting the contexts among neighbor keys, Remove Query exhibits better performances than Remove Key. The concatenation of contextualized key and query (i.e., the run of CoT block with 3×3 convolution) leads to performance boosts, which validate the complementarity of the contextualized key and query for measuring dynamic context.

4.1.6 Effect of Replacement Settings

In order to show the relationship between performance and the number of stages replaced with our CoT blocks, we progressively replace the stages with our CoT blocks in ResNet-50 backbone (res2→res3→res4→res5), and compare the performances. The results shown in Table 7 indicate that increasing the number of stages replaced with CoT blocks can generally lead to performance improvement, and meanwhile the parameter number & FLOPs are slightly decreased. When taking a close look on the throughputs and accuracies of different replacement settings, the replacement of CoT

TABLE 7
Effect of Utilizing Different Replacement Settings on the Four Stages (**res2→res3→res4→res5**) in the Basic Backbone of ResNet-50 and Two Widely Adopted Architecture Changes, ResNet-D [52] and Squeeze-and-Excitation [23] (**D-SE**)

	res2	res3	res4	res5	D-SE	Params	GFLOPs	Memory	Infer	Top-1 Acc.	Top-5 Acc.
ResNet-50						25.5M	4.1	2810.10 MiB	508 ex/s	77.3	93.6
CoTNet-50				✓		23.5M	4.0	2893.12 MiB	491 ex/s	78.5	94.1
			✓	✓		22.4M	3.7	2884.12 MiB	443 ex/s	79.0	94.3
		✓	✓	✓		22.3M	3.4	2882.88 MiB	390 ex/s	79.0	94.4
	✓	✓	✓	✓		22.2M	3.3	2882.71 MiB	331 ex/s	79.2	94.5
SE-ResNetD-50					⋆	35.7M	4.4	2848.63 MiB	444 ex/s	79.1	94.5
SE-CoTNetD-50			✓	✓	⋆	23.1M	4.1	2891.09 MiB	414 ex/s	79.8	94.7

✓ *denotes the stage is replaced with our CoT blocks.* ⋆ *denotes the use of architecture changes (D-SE). We adopt the default setup for training on ImageNet and the batch size at inference is set as 64.*

TABLE 8
Performance Comparisons with the State-of-the-Art Vision Backbones on the Downstream Task of Object Detection (Base Detectors: Faster-RCNN and Cascade-RCNN)

	Backbone	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Faster-RCNN	ResNet-50 [3]	39.34	59.47	42.76	23.57	42.42	51.30
	ResNeXt-50 [20]	41.31	62.23	44.91	25.33	44.52	53.20
	ResNeSt-50 [27]	42.39	63.73	46.02	26.25	45.88	54.24
	CoTNet-50	**43.50**	**64.84**	**47.53**	**26.36**	**47.54**	**56.49**
	CoTNeXt-50	**44.06**	**65.76**	**47.65**	**27.08**	**47.70**	**57.21**
	SE-CoTNetD-50	**43.96**	**65.20**	**48.25**	**27.71**	**47.05**	**56.51**
	ResNet-101 [3]	41.46	61.99	45.38	25.31	44.75	54.62
	ResNeXt-101 [20]	42.91	63.77	46.89	25.96	46.42	55.47
	ResNeSt-101 [27]	44.13	61.91	47.67	26.02	47.69	57.48
	CoTNet-101	**45.35**	**66.80**	**49.18**	**28.65**	**49.47**	**58.82**
	CoTNeXt-101	**46.10**	**67.50**	**50.22**	**29.44**	**49.84**	**59.26**
	SE-CoTNetD-101	**45.66**	**66.86**	**50.11**	**29.83**	**49.25**	**59.17**
Cascade-RCNN	ResNet-50 [3]	42.45	59.76	46.09	24.90	45.64	55.86
	ResNeXt-50 [20]	44.53	62.45	48.38	27.29	48.01	57.87
	ResNeSt-50 [27]	45.41	63.92	48.70	28.77	48.69	58.43
	CoTNet-50	**46.11**	**64.68**	**49.75**	28.71	**49.76**	**60.28**
	CoTNeXt-50	**46.79**	**65.54**	**50.53**	**29.74**	**50.49**	**61.04**
	SE-CoTNetD-50	**46.77**	**64.91**	**50.46**	**28.90**	**50.28**	**60.92**
	ResNet-101 [3]	44.13	61.91	47.67	26.02	47.69	57.48
	ResNeXt-101 [20]	45.83	63.61	49.89	27.75	49.53	59.14
	ResNeSt-101 [27]	47.51	66.06	51.35	30.25	50.96	61.23
	CoTNet-101	**48.19**	**67.00**	**52.17**	30.00	**52.32**	**62.87**
	CoTNeXt-101	**49.02**	**67.67**	**53.03**	**31.44**	**52.95**	**63.17**
	SE-CoTNetD-101	**49.02**	**67.78**	**53.15**	**31.26**	**52.76**	**63.29**

Average Precision (AP) is reported at different IoU thresholds and for three different object sizes: small, medium, large (s/m/l).

blocks in the last two stages (res4 and res5) contributes to the most performance boost. The additional replacement of CoT blocks in the first stages (res2 and res3) can only lead to a marginal performance improvement (0.2% top-1 accuracy in total), while requiring 1.34× inference time. In addition, when only replacing the last stage (res5) in ResNet-50 with our CoT block, the memory cost slightly increases. The additional replacement of CoT blocks in the first three stages (res2, res3, and res4) does not make a major difference in memory cost. Therefore, in order to seek a better speed-accuracy trade-off, we follow [44] and construct an upgraded version of our CoTNet, named SE-CoTNetD-50, where only the 3×3 convolutions in the res4 and res5 stages are replaced with CoT blocks under SE-ResNetD-50 backbone. Note that the SE-ResNetD-50 backbone is a variant of ResNet-50 with two widely adopted architecture changes (ResNet-D [52] and Squeeze-and-Excitation in all bottleneck blocks [23]). As shown in Table 7, compared to the SE-ResNetD-50 counterpart, our SE-CoTNetD-50 achieves better performances at a virtually negligible decrease in throughput.

4.2 Object Detection

4.2.1 Setup

We next evaluate the pre-trained CoTNet for the downstream task of object detection on COCO dataset, which aims to localize and recognize objects at bounding-box level. For this task, we adopt Faster-RCNN [55], [56] and Cascade-RCNN [57] as the base object detectors, and directly replace the vanilla ResNet backbone with our CoTNet. Following the standard setting in [20], we train all models on COCO-2017 training set (∼118K images) and evaluate them on COCO-2017 validation set (5K images). The standard AP metric of single scale is adopted for evaluation. During training, for each input image, the size of the shorter side is sampled from the range of [640, 800]. All models are trained with FPN [58] and synchronized batch normalization [59]. We utilize the 1x learning rate schedule for training. For fair comparison with other vision backbones in this task, we set all the hyperparameters and detection heads as in [27].

4.2.2 Performance Comparison

Table 8 summarizes the performance comparisons on COCO dataset for object detection by leveraging Faster-RCNN and Cascade-RCNN in different pre-trained backbones. We group the vision backbones with same network depth (50-layer/101-layer). From observation, our pre-trained CoTNet models (CoTNet-50/101 and CoTNeXt-50/101) exhibit a clear performance boost against the ConvNets backbones (ResNet-50/101 and ResNeSt-50/101) for each network depth across all IoU thresholds and object sizes. The results basically demonstrate the advantage of integrating self-attention learning with contextual information mining in CoTNet, even when transferred to the downstream task of object detection.

4.3 Instance Segmentation

4.3.1 Setup

Here we evaluate the pre-trained CoTNet in another downstream task of instance segmentation on COCO dataset. This task goes beyond the box-level understanding in object detection by additionally predicting the object mask for each detected object, pursuing the pixel-level understanding of visual content. Specifically, Mask-RCNN [60], [61] and Cascade-Mask-RCNN [57] are utilized as the base models for instance segmentation. In the experiments, we replace the vanilla ResNet backbone in Mask-RCNN with our CoTNet. Similarly, all models are trained with FPN and synchronized batch normalization. We adopt the 1x learning rate schedule during training, and all the other hyperparameters are set as in [27]. For evaluation, we report the standard COCO metrics including both bounding box and mask AP (i.e., AP^{bb} and AP^{mk}).

4.3.2 Performance Comparison

Table 9 details the performances of Mask-RCNN with different pre-trained vision backbones for the downstream task of instance segmentation on COCO dataset. Similar to the observations for object detection downstream task, our pre-trained CoTNet models yields consistent gains against both ConvNets backbones (ResNet-50/101 and ResNeSt-50/101) and attention-based model (BoTNet-50/101) over the most IoU thresholds. This generally highlights the generalizability of our CoTNet in the challenging instance segmentation task. In particular, BoTNet-50 achieves better performances than the best ConvNets (ResNeSt-50). This might attribute to the additional modeling of global self-attention in BoTNet plus the more advanced fine-tuning setup with larger input size (1024×1024) and longer training epochs (36). However, by uniquely exploiting the contextual information among neighbor keys for self-attention

TABLE 9
Performance Comparisons with the State-of-the-Art Vision Backbones on the Downstream Task of Instance Segmentation (Base Models: Mask-RCNN and Cascade-Mask-RCNN)

	Backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
Mask-RCNN	ResNet-50 [3]	39.97	60.19	43.73	36.05	57.02	38.54
	ResNeXt-50 [20]	41.74	62.32	45.60	37.41	59.24	39.98
	ResNeSt-50 [27]	42.81	63.93	46.85	38.14	60.54	40.69
	BoTNet-50 [31]	43.6	65.3	47.6	38.9	62.5	41.3
	CoTNet-50	**44.06**	64.99	**48.29**	**39.28**	62.12	**42.17**
	CoTNeXt-50	**44.47**	**65.74**	**48.71**	**39.62**	**62.70**	**42.35**
	SE-CoTNetD-50	**44.16**	65.26	**48.32**	**39.38**	62.18	**42.23**
	ResNet-101 [3]	41.78	61.90	45.80	37.50	58.78	40.21
	ResNeXt-101 [20]	43.25	63.61	47.23	38.60	60.74	41.37
	ResNeSt-101 [27]	45.75	66.88	49.75	40.65	63.76	43.68
	BoTNet-101 [31]	45.5	-	-	40.4	-	-
	CoTNet-101	**46.17**	**67.17**	**50.63**	**40.86**	**64.18**	43.64
	CoTNeXt-101	**46.66**	**67.70**	**50.90**	**41.21**	**64.45**	**44.27**
	SE-CoTNetD-101	**46.67**	**67.85**	**51.30**	**41.53**	**64.92**	**44.69**
Cascade-Mask-RCNN	ResNet-50 [3]	43.06	60.29	46.55	37.19	57.61	40.01
	ResNeXt-50 [20]	44.91	62.66	48.80	38.57	59.83	41.59
	ResNeSt-50 [27]	46.23	64.62	50.15	39.64	61.86	42.88
	CoTNet-50	**46.94**	**65.36**	**50.69**	**40.25**	**62.37**	**43.38**
	CoTNeXt-50	**47.63**	**65.93**	**51.64**	**40.76**	**63.32**	**44.01**
	SE-CoTNetD-50	**47.44**	**65.93**	**51.27**	**40.73**	**63.22**	**44.09**
	ResNet-101 [3]	44.79	62.31	48.46	38.51	59.33	41.53
	ResNeXt-101 [20]	46.24	64.01	49.92	39.77	61.19	43.06
	ResNeSt-101 [27]	48.44	66.80	52.60	41.52	64.03	45.02
	CoTNet-101	**48.97**	**67.42**	**53.10**	**41.98**	**64.81**	**45.39**
	CoTNeXt-101	**49.35**	**67.88**	**53.53**	**42.20**	**65.00**	**45.69**
	SE-CoTNetD-101	**49.24**	**67.45**	**53.36**	**42.38**	**64.79**	**45.89**

The bounding box and mask Average Precision (AP^{bb}, AP^{mk}) are reported at different IoU thresholds. Note that BoTNet-50/101 is fine-tuned with larger input size 1024×1024 and longer epochs (36).

TABLE 10
Performance Comparisons with the State-of-the-Art Methods under Different Vision Backbones for the Downstream Task of Semantic Segmentation on the ADE20K Validation Set

Method	Backbone	mIoU
UPerNet [62]	ResNet-50	42.8
DNL [63]	ResNet-50	43.0
PSPNet [64]	ResNet-50	43.4
FCN [26]	HRNetV2p-W48	43.9
DMNet [65]	ResNet-50	44.2
DeeplabV3 [66]	ResNet-50	44.1
DeeplabV3 [27]	ResNeSt-50	45.1
UPerNet [67]	DeiT-S	43.8
UPerNet	CoTNet-50	**46.4**
UPerNet	CoTNeXt-50	**46.5**
UPerNet	SE-CoTNetD-50	**46.7**
UPerNet [62]	ResNet-101	44.9
PSPNet [64]	ResNet-101	45.4
DNL [63]	ResNet-101	45.8
DMNet [65]	ResNet-101	46.8
DeeplabV3 [66]	ResNet-101	46.7
DeeplabV3 [27]	ResNeSt-101	46.9
UPerNet [67]	DeiT-B	47.2
UPerNet	CoTNet-101	**47.8**
UPerNet	CoTNeXt-101	**47.3**
UPerNet	SE-CoTNetD-101	**49.0**

The mean IoU (mIoU) averaged over all classes is reported for evaluation.

learning, our CoTNet-50 manages to lead the performance boosts over the most metrics, even when fine-tuned with smaller input size and less epoches (12). The results again confirm the merit of simultaneously performing context mining and self-attention learning in our CoTNet for visual representation learning.

4.4 Semantic Segmentation

4.4.1 Setup

We finally conduct the evaluation of our pre-trained CoTNet over the downstream task of semantic segmentation on ADE20K dataset. Different from instance segmentation that predicts the label of each pixel within each instance uniquely, semantic segmentation labels every pixel without instance-level discrimination. ADE20K is a commonly adopted benchmark for semantic segmentation, which covers 150 semantic categories (i.e., 35 stuff classes and 115 discrete object classes). This dataset consists of 25K images, including 20K images for training, 2K images for validation, and 3K images for testing. In the experiments, we adopt UPerNet [62] as the base model for semantic segmentation downstream task. The vanilla ResNet backbones (ResNet-50/ResNet-101) in UPerNet are directly replaced with our CoTNet. Models are trained on 8 GPUs with 2 images per GPU for 160K iterations, and we further evaluate them over the ADE20K validation set. The metric of mean IoU (mIoU) averaged over all classes is reported for evaluation.

4.4.2 Performance Comparison

Table 10 shows the detailed performance comparisons on the ADE20K validation set for semantic segmentation by capitalizing on the state-of-the-art methods in different pre-trained backbones. We group all runs with similar network depth (50-layer/101-layer) or parameter budget. Similarly, when utilizing the same semantic segmentation approach (i.e., UPerNet), our pre-trained CoTNet models (CoTNet-50/101) achieve a clear performance improvement than the ConvNets backbones (ResNet-50/101) for each network depth. Furthermore, by upgrading CoTNet-101 with two architecture changes (ResNet-D and Squeeze-and-Excitation in all bottleneck blocks), our SE-CoTNetD-101 can reach 49.0% in mIoU and make the absolute improvement over the best competitor DeiT-B by 1.8%, which clearly demonstrates the superiority of CoTNet for semantic segmentation downstream task.

5 Conclusion

In this work, we propose a new Transformer-style architecture, termed Contextual Transformer (CoT) block, which exploits the contextual information among input keys to guide self-attention learning. CoT block first captures the static context among neighbor keys, which is further leveraged to trigger self-attention that mines the dynamic context. Such way elegantly unifies context mining and self-attention learning into a single architecture, thereby strengthening the capacity of visual representation. Our CoT block can readily replace standard convolutions in existing ResNet architectures, meanwhile retaining the favorable parameter budget. To verify our claim, we construct Contextual Transformer Networks (CoTNet) by replacing the 3×3 convolutions in ResNet architectures (e.g., ResNet or ResNeXt). The CoTNet architectures learnt on ImageNet validate our proposal and analysis. Extensive experiments conducted on COCO and ADE20K datasets in the context of object detection, instance segmentation, and semantic segmentation also demonstrate the generalization of the visual representation pre-trained by our CoTNet over various downstream tasks.

References

[1] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1251–1258.

[2] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 764–773.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Informat. Process. Syst.*, 2012, pp. 1097–1105.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015.

[6] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[7] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[8] R. Mottaghi *et al.*, "The role of context for object detection and semantic segmentation in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 891–898.

[9] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, "Objects in context," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, 2007, pp. 1–8.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2019, pp. 4171–4186.

[11] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Informat. Process. Syst.*, 2017, pp. 5998–6008.

[12] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, "Attention augmented convolutional networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3286–3295.

[13] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 213–229.

[14] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations*, 2020.

[15] Y. Li, Y. Pan, T. Yao, J. Chen, and T. Mei, "Scheduled sampling in vision-language pretraining with decoupled encoder-decoder network," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 8518–8526.

[16] Y. Pan, T. Yao, Y. Li, and T. Mei, "X-linear attention networks for image captioning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10 971–10 980.

[17] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, "Stand-alone self-attention in vision models," in *Proc. Adv. Neural Informat. Process. Syst.*, 2019, pp. 68–80.

[18] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10 076–10 085.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[20] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1492–1500.

[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.

[22] M. Jaderberg *et al.*, "Spatial transformer networks," in *Proc. Adv. Neural Informat. Process. Syst.*, 2015, pp. 2017–2025.

[23] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.

[24] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 2011–2023, Aug. 2020.

[25] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2net: A new multi-scale backbone architecture," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 2, pp. 652–662, Feb. 2021.

[26] J. Wang *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020, pp. 5686–5696.

[27] H. Zhang *et al.*, "Resnest: Split-attention networks," 2020, *arXiv: 2004.08955*.

[28] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7794–7803.

[29] H. Hu, Z. Zhang, Z. Xie, and S. Lin, "Local relation networks for image recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3464–3473.

[30] M. Chen *et al.*, "Generative pretraining from pixels," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1691–1703.

[31] A. Srinivas, T.-Y. Lin, N. Parmar, J. Shlens, P. Abbeel, and A. Vaswani, "Bottleneck transformers for visual recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 16 519–16 529.

[32] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," *Proc. Adv. Neural Informat. Process. Syst.*, vol. 34, pp. 15908–15919, 2021.

[33] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 10 012–10 022.

[34] X. Chu *et al.*, "Twins: Revisiting the design of spatial attention in vision transformers," *Proc. Adv. Neural Informat. Process. Syst.*, vol. 34, pp. 9355–9366, 2021.

[35] A. Ali *et al.*, "XCiT: Cross-covariance image transformers," in *Adv. Neural Informat. Process. Syst.*, vol. 34, pp. 20014–20027, 2021.

[36] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 510–519.

[37] D. Haase and M. Amthor, "Rethinking depthwise separable convolutions: How intra-kernel correlations lead to improved mobilenets," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 14 600–14 609.

[38] X. Wang and X. Y. Stella, "Tied block convolution: Leaner and better CNNs with shared thinner filters," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 10 227–10 235.

[39] J. Chen, X. Wang, Z. Guo, X. Zhang, and J. Sun, "Dynamic region-aware convolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8064–8073.

[40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[41] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[42] B. Zhou *et al.*, "Semantic understanding of scenes through the ade20k dataset," *Int. J. Comput. Vis.*, vol. 127, no. 3, pp. 302–321, 2019.

[43] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," 2016, *arXiv:1608.03983*.

[44] I. Bello, "Lambdanetworks: Modeling long-range interactions without attention," in *Proc. Int. Conf. Learn. Representations*, 2020.

[45] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 702–703.

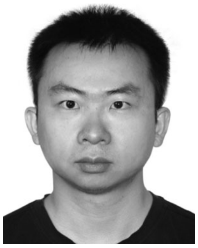
[46] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. Int. Conf. Learn. Representations*, 2018.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[48] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1058–1066.

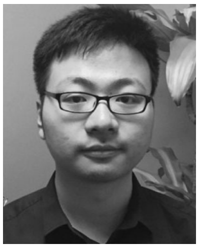
[49] W. Xu, Y. Xu, T. Chang, and Z. Tu, "Co-scale conv-attentional image transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 9981–9990.

[50] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, "Going deeper with image transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 32–42.
[51] S. dAscoli, H. Touvron, M. L. Leavitt, A. S. Morcos, G. Biroli, and L. Sagun, "ConViT: Improving vision transformers with soft convolutional inductive biases," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 2286–2296.
[52] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 558–567.
[53] I. Bello *et al.*, "Revisiting resnets: Improved training and scaling strategies," 2021, *arXiv:2103.07579*.
[54] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu, "GCnet: Non-local networks meet squeeze-excitation networks and beyond," in *Proc. IEEE/CVF Int. Conf. on Comput. Vis. Workshops*, 2019, pp. 1971–1980.
[55] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
[56] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Informat. Process. Syst.*, 2015, pp. 91–99.
[57] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6154–6162.
[58] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2117–2125.
[59] H. Zhang *et al.*, "Context encoding for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7151–7160.
[60] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.
[61] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 386–397, Feb. 2020.
[62] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 432–448.
[63] M. Yin, Z. Yao, Y. Cao, X. Li, Z. Zhang, S. Lin, and H. Hu, "Disentangled non-local neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 191–207.
[64] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2017, pp. 2881–2890.
[65] J. He, Z. Deng, and Y. Qiao, "Dynamic multi-scale filters for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3562–3572.
[66] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 833–851.
[67] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 10 347–10 357.



Yehao Li received the PhD degree from Sun Yet-sen University (SYSU), Guangzhou, China, in 2019. He is currently a researcher with Vision and Multimedia Lab, JD Explore Academy, Beijing, China. He was one of core designers of top-performing multimedia analytic systems in worldwide competitions such as COCO Image Captioning, ActivityNet Dense-Captioning Events in Videos Challenge 2017 and Visual Domain Adaptation Challenge 2018. His research interests include vision and language, large-scale visual search, and video understanding.

Ting Yao (Senior Member, IEEE) is currently a principal researcher with Vision and Multimedia Lab, JD Explore Academy, Beijing, China. His research interests include video understanding, vision and language, and deep learning. Prior to joining JD.com, he was a researcher with Microsoft Research Asia, Beijing, China. Ting is the principal designer of several top-performing multimedia analytic systems in international benchmark competitions such as ActivityNet Large Scale Activity Recognition Challenge 2019–2016, Visual Domain Adaptation Challenge 2019–2017, and COCO Image Captioning Challenge. He is the leader organizer of MSR Video to Language Challenge in ACM Multimedia 2017 & 2016, and built MSR-VTT, a large-scale video to text dataset that is widely used worldwide. His works have led to many awards, including ACM SIGMM Outstanding PhD Thesis Award 2015, ACM SIGMM Rising Star Award 2019, and IEEE TCMC Rising Star Award 2019. He is also an associate editor of IEEE Transactions on Multimedia.



Yingwei Pan (Member, IEEE) received the PhD degree in electrical engineering from the University of Science and Technology of China, in 2018. He is currently a researcher with Vision and Multimedia Lab, JD Explore Academy, Beijing, China. His research interests include vision and language, domain adaptation, and large-scale visual search. He was one of core designers of top-performing multimedia analytic systems in worldwide competitions such as COCO Image Captioning, Visual Domain Adaptation Challenge 2018, ActivityNet Dense-Captioning Events in Videos Challenge 2017, and MSR-Bing Image Retrieval Challenge 2014 and 2013.

Tao Mei (Fellow, IEEE) is a vice president with JD.COM and the deputy managing director of JD Explore Academy, where he also serves as the director of Computer Vision and Multimedia Lab. Prior to joining JD.COM in 2018, he was a senior research manager with Microsoft Research Asia in Beijing, China. He has authored or coauthored more than 200 publications (with 12 best paper awards) in journals and conferences, 10 book chapters, and edited five books. He holds more than 25 U.S. and international patents. He is or has been an editorial board member of *IEEE Transactions on Image Processing*, *IEEE Transactions on Circuits and Systems for Video Technology*, *IEEE Transactions on Multimedia*, *ACM Transactions on Multimedia Computing, Communications, and Applications*, *Pattern Recognition*, etc. He is the general co-chair of IEEE ICME 2019, the Program Co-chair of ACM Multimedia 2018, IEEE ICME 2015 and IEEE MMSP 2015. He is a fellow of IAPR (2016), a distinguished scientist of ACM (2016), and a distinguished Industry Speaker of IEEE Signal Processing Society (2017).