Grammatically Recognizing Images with Tree Convolution

Guangrun Wang[1,3], Guangcong Wang[1], Keze Wang[2,3], Xiaodan Liang[1,3], Liang Lin[1,3*]
[1]Sun Yat-sen University, [2]University of California, Los Angeles, [3]DarkMatter AI Research
{wanggrun, wanggc3}@mail2.sysu.edu.cn, {kezewang, xdliang328}@gmail.com, linliang@ieee.org

ABSTRACT

Similar to language, understanding an image can be considered as a hierarchical decomposition process from scenes to objects, parts, pixels, and the corresponding spatial/contextual relations. However, the existing convolutional networks concentrate on stacking redundant convolutional layers with a large number of kernels in a hierarchical organization to implicitly approximate this decomposition. This may limit the network to learn the semantic information conveyed in the internal feature maps that may reveal minor yet crucial differences for visual understanding. Attempting to tackle this problem, this paper proposes a simple yet effective tree convolution (TreeConv) operation for deep neural networks. Specifically, inspired by the image grammar techniques [73] that serve as a unified framework of object representation, learning, and recognition, our TreeConv designs a generative image grammar, i.e., tree generation rule, to parse the hierarchy of internal feature maps by generating tree structures and implicitly learning the specific visual grammars for each object category. Extensive experiments on a variety of benchmarks, i.e., classification (ImageNet / CIFAR), detection & segmentation (COCO 2017), and person re-identification (CUHK03), demonstrate the superiority of our TreeConv in both boosting the accuracy and reducing the computational cost. The source code will be available at https://github.com/wanggrun/TreeConv.

KEYWORDS

Deep neural network architecture, hierarchical representation learning, image grammar, image classification, object detection, person re-identification

ACM Reference Format:
Guangrun Wang[1,3], Guangcong Wang[1], Keze Wang[2,3], Xiaodan Liang[1,3], Liang Lin[1,3]. 2020. Grammatically Recognizing Images with Tree Convolution. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403133

1 INTRODUCTION

Achieving remarkable success on various vision tasks, deep convolutional neural networks (DNNs) have been widely studied and

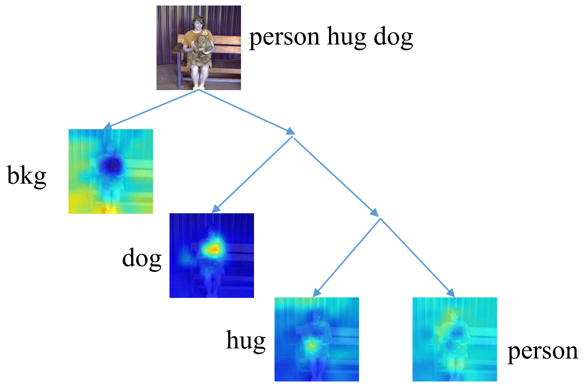

Figure 1: An example parse tree of grammatically recognizing images. As shown, the parse tree covers a variety of visual concepts, including objects (i.e., dog and person), actions (i.e., hug), and background (i.e., "bkg"), which accord with human perception. The corresponding heatmap denotes the parsed visualization.

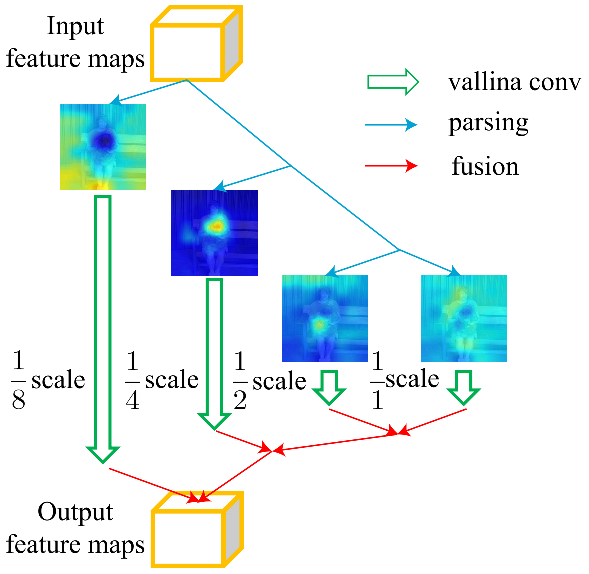

Figure 2: An overview of our TreeConv operation. Given the input feature maps, our TreeConv first generates constituency-based parse trees, and then performs convolution with redundancy reduction for different nodes in the corresponding scales. The output feature maps are obtained by fusing the above convolved results.

improved by introducing more powerful architectures (e.g., ResNet [15]) and effective layers (e.g., [6]). However, most of these works still rely on the standard convolutional operation, i.e., vanilla convolution, which regards the input image / feature maps within each DNN layer as 2D-sequential signals for processing through a sliding-window fashion. Therefore, all the regions of input feature maps are supposed to be indistinguishable and convolved by

[*]The corresponding author is Liang Lin.

the same convolutional kernel. This operation may ignore the underlying structure within the input image / feature map, which is conceptually hierarchical rather than strictly sequential.

To overcome this limitation, sufficient convolutional layers with a large number of hidden neurons need to be stacked up to implicitly encode the hierarchical structure of the input image. However, this is inconsistent with human perceptions, and also inefficiency in capturing long-term dependencies as well as containing spatial redundancies [2, 3].

According to the theory of image grammar [73], the semantic content of the input image / feature map can be represented / generated by a set of grammar rules, i.e., visual grammars, which is a powerful tool to model the high-level human knowledge for specific domains in a hierarchical and compositional manner. These domains include the decompositions of scenes [12], semantic relations between humans and objects [65], and dependency of human parts with attributes [58]. Fig. 1 is a typical example of how a scene image (sampled from ImageNet [39]) can be parsed into a hierarchical tree with nodes representing regions (e.g., a dog, a person, and a region of interaction "hugging") and edges representing the relation between different regions. As pointed out by the existing theories [25, 27, 45, 46, 63], this hierarchical tree structure also exists in the internal feature map and is considered to be beneficial for being integrated to enhance the network capacity of deep models.

Due to the lack of annotations, reconciling the convolutional operation finely with the tree structure generated from human perceptions is a fundamental yet challenging problem. Intuitively, one straightforward way is to manually annotate a hierarchical structure (e.g., tree) for each image as a guide. However, this is infeasible for the following two reasons: i) the annotation is extremely expensive; ii) the annotation usually has an ambiguity, i.e., an image may have more than one way of hierarchical decomposition.

Attempting to overcome the limitation of the convolutional operation and implicitly learn the visual grammar for decomposing the input image / feature map into a hierarchical structural representation, we propose a novel tree convolution (TreeConv) operation for improving the CNNs via the guidance of visual grammars within each DNN layer. Specifically, our TreeConv includes three main steps: **i)** parsing the input feature maps by using trees via a top-down parsing; **ii)** resizing the tree nodes to different scales for redundancy reduction and convolving them by vanilla convolutions; **iii)** fusing the convolved nodes from the bottom up to form the output feature maps. Fig. 2 illustrates the above three steps. According to our observation, the hierarchical features seem to be ordered regularly, i.e., they are sorted by the fineness, implying there may be some knowledge of visual grammar implicitly learned by our model. Moreover, as a by-product, the automatically sorted hierarchical features enable us to reduce the redundancy of the input feature maps and thus greatly reduce the computational cost, compared with the vanilla convolution.

Overall, the main contributions are two-fold. **First**, we propose a novel convolutional operation named TreeConv to enable the ability of implicitly learning semantic information (i.e., visual grammars) about hierarchical human perception within each DNN layer. This is beneficial for alleviating the problems raised by CNNs, such as inconsistency with human perception and inefficiency in capturing long-term dependencies. **Second**, thanks to the good property of hierarchical convolution, our TreeConv is able to reduce the network redundancy of existing state-of-the-art network architectures while improving their discriminative power. Extensive experimental results on a variety of benchmarks (i.e., ImageNet classification, COCO 2017 detection and segmentation, CUHK03 person re-identification, and CIFAR recognition) demonstrate the superiority of our TreeConv.

2 RELATED WORK

Trees in Neural Networks: Existing works suggest that introducing structure information into CNNs is beneficial for many machine learning tasks, such as image classification [38], semantic segmentation [27, 63], object detection [45, 46], language processing [35], knowledge discovery and data mining [23, 72], and general AI systems [10]. However, these works have two drawbacks. First, the structures they use are handcrafted and fixed. Second, they only consider the hierarchy of input images while ignoring the hierarchy of internal feature maps. Beyond CNNs and computer vision, many works on grammar induction are based on RNNs and natural language modeling. Similar to that in CNNs, recent results suggest that introducing structure information into RNNs is also beneficial [30, 41, 59]. For example, [42] showed that LSTMs with tree structures perform better than standard LSTMs in a wide range of language tasks.

Visual Grammar: As a topic of interest in neural language for many years, grammar models [12, 58, 65, 73] have been applied to solving the computer vision problems for their expressive capability and human-perception consistency in modeling structures and relations of semantic contents inside the input image. Briefly, the grammar models can be coarsely divided into two variations: phase structure grammar (PG) and dependency grammar (DG). In the PG, the constituency relation is defined to ensure each node must geometrically contain all of its constituents for representing the compositional structures [9]. However, it has difficulty in handling large deformations of objects. In the DG, constituent parts do not need to be contained within their parents but instead are constrained by an adjacency relation [13]. The DG advances in representing objects with large articulated deformations, but cannot support the coarse-to-fine summarization. The recursive cortical networks [11] have been proposed with better data efficiency in learning, which adopts the AND-OR grammar framework [73]. The recent proposed AOGNet [25] advances existing works by imposing compositional grammatical architectures for deep learning and presents deep AND-OR Grammar networks. However, AOGNet has several drawbacks, including handcrafted grammar, debatable tree nodes, and the unexplainable channel-wise grammatical rules. Please see Section 3.5 for more details.

Reducing Network Redundancy: To improve the network efficiency, most of the existing works focus on improving the connectivity [15, 21, 75], reducing the channels [18, 40], and using depth-wise/group convolutions [5, 64]. Some recent works start to consider removing spatial redundancy [2, 3]. However, these methods manually define spatial redundancy. In contrast, the redundancy is naturally obtained as a by-product in our TreeConv, thanks to the good property of our implicitly learned visual grammars.

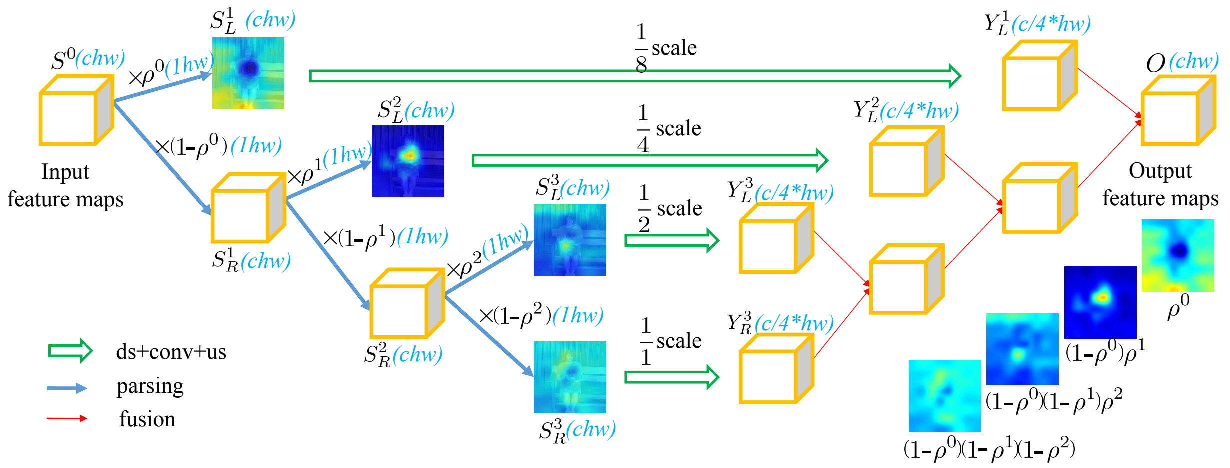

Figure 3: Detail of our TreeConv. ds: downsampling; us: upsampling; conv: 3×3 convolution. x-scale: The feature maps are first downsampled by x times and then convolved by vanilla convolution before being upsampled by $1/x$ times. chw, 1hw, c/4*hw: shapes of the feature maps. Here, ρ^0, ρ^1, and ρ^2 are obtained by: $\rho^0 = \frac{1}{1+\exp(W^0S^0)}$, $\rho^1 = \frac{1}{1+\exp(W^1S^1)}$, $\rho^2 = \frac{1}{1+\exp(W^2S^2)}$, respectively.

Attention Based Model: Through capturing the long dependencies by focusing on the most important part of the input data, attention-based models and gate mechanism (e.g., LSTM [17] and gated convolution [60]) have achieved remarkable success in various artificial intelligence and data mining tasks, e.g., machine translation [48], graph embedding [8], generative modeling [62], visual recognition [19, 20, 53, 55], and knowledge discovery and data mining [32, 47, 56, 71]. In contrast to these proposed attention mechanisms, our proposed TreeConv considers the input data as a whole for grammatical modeling within each DNN layer to implicitly capture the long-range dependencies of input data via a hierarchical decomposition fashion. This is beneficial for alleviating the problems raised by attention mechanisms, such as inconsistency with human perception and inefficiency in capturing long-term dependencies.

3 TREE CONVOLUTION

3.1 A Rule of Tree Generation

Instead of predefining concrete visual grammars during the network initialization, we design a generative image grammar (i.e., tree generation rule) to implicitly learn the specific visual grammars for each object category from sufficient training data (e.g., ImageNet) in a purely data-driven manner. Inspired by hierarchical, compositional, and reconfigurable with lateral connections of the phrase structure grammar [9], the tree generation rule is defined as follows:

Definition. (*Tree Generation Rule*) Let S^0 denote a root node, which is fed with input feature maps. Then, a constituency-based parse tree is generated by:

1. $S^0 \to S_L^1, S^0 \to S_R^1$, where the left child S_L^1 is a terminal node which is large-region, informationally redundant, and contextually global.

2. $S_R^1 \to S_L^2, S_R^1 \to S_R^2$, where the left child S_L^2 is also a terminal node which is more informationally redundant and more contextually global than S_R^2.

3. $S_R^2 \to S_L^3, S_R^2 \to S_R^3$. Both S_L^3 and S_L^3 are terminal nodes. S_R^3 is most informative in all nodes, usually capturing the finest details of the input feature maps.

Flexible visual grammars. By using the predefined *tree generation rule*, a tree is generated. Although the structure of this tree is handcrafted and fixed during the training and testing, the nodes of this tree can be connected to different regions for different input feature maps in an adaptive, flexible, and personalized way. More importantly, we do not impose any predefined concept on these tree nodes. We argue that each node of the tree may implicitly learn to represent a semantic concept for each object category. As in Fig. 4, for the category "dog", S_L^1 represents the background; S_L^2 represents the dog head; S_L^2 means the dog body; S_L^3 is the dog mouth. While for the category "bottle", the tree nodes have completely different meanings. Hence, although the main structure of the tree is always fixed, the learned visual grammars are capable of being flexible and implicit for different object categories.

Increasing tree depth. Our definition empirically sets the depth of the tree as three. Actually, S_R^3 can be further parsed into child nodes to increase the tree depth. However, according to the definition, the information in S_R^3 is fine and detailed. However, parsing S_R^3 further (i.e., increasing the tree depth) may lead to the over-parsing of feature maps, while reducing tree depth may lead to the under-parsing of feature maps.

Our TreeConv operation consists of three steps (see Fig. 3), i.e., **i)** parsing the input feature maps by using the trees via a top-down manner, **ii)** reducing the redundancy of the nodes by convolving them in different scales, and **iii)** fusing the nodes via a bottom-up fashion to form the output feature maps. Please see Section 3.2 - 3.4 for more details.

3.2 Parsing Feature Maps

As shown in Fig. 3, the parsing process starts with the root node $S^0 \in \mathcal{R}^{c\times h\times w}$, which is exactly the input feature maps. S^0 is first parsed into a left child $S_L^1 \in \mathcal{R}^{c\times h\times w}$ and a right child $S_R^1 \in \mathcal{R}^{c\times h\times w}$. According to the *tree generation rule*, the left child S_L^1 occupies the dominant area of S_0. To model this property, we use ρ^0 to denote the component ratio of S_L^1 in S^0 and define ρ^0 as:

$$\rho^0 = \frac{1}{1+\exp(W^0 S^0)}, \tag{1}$$

where $W^0 \in \mathcal{R}^{1\times c}$, $S^0 \in \mathcal{R}^{c\times(hw)}$, and $\rho^0 \in \mathcal{R}^{1\times(hw)}$. Actually, Eq. (1) can be implemented by using a simple 1×1 vanilla convolution followed by a sigmoid activation function. With the definition of ρ^0, the left child S_L^1 and right child S_R^1 of S^0 can be computed by:

$$S_L^1 = \rho^0 S^0,\ S_R^1 = S^0 - S_L^1. \tag{2}$$

By analogy, S_R^1 can be further parsed into two child nodes S_R^2 and S_L^2 by defining $\rho^1 = \frac{1}{1+\exp(W^1 S_R^1)}$ where $W^1 \in \mathcal{R}^{1\times c}$ and $S_R^1 \in \mathcal{R}^{c\times(hw)}$. We have $S_L^2 = \rho^1 S_R^1, S_R^2 = S_R^1 - S_L^2$. Furthermore, S_R^2 can be parsed into two child nodes S_R^3 and S_L^3 by defining $\rho^2 = \frac{1}{1+\exp(W^2 S_R^2)}$ where $W^2 \in \mathcal{R}^{1\times c}$ and $S_R^2 \in \mathcal{R}^{c\times(hw)}$. We have $S_L^3 = \rho^2 S_R^2, S_R^3 = S_R^2 - S_L^3$. Fig. 3 illustrates the process of parsing the input feature maps by using a tree. The parsing has two good properties, i.e., *normalization* and *orderliness*.

Normalization. The parsed nodes sum up to the input feature maps, i.e., $S^0 = S_L^1 + S_L^2 + S_L^3 + S_R^3$. This property can be proved by: $S_L^1 + S_L^2 + (S_L^3 + S_R^3) = S_L^1 + (S_L^2 + S_R^2) = (S_L^1 + S_R^1) = S^0$. The normalization property is in favor of the *tree generation rule* by guaranteeing that there is no information loss during the parsing process.

Orderliness. As the tree depth increases, the effective region of the right children becomes smaller and smaller. This property can be proved by the fact $S_R^d = (1-\rho^0)(1-\rho^1)\cdots(1-\rho^{d-1})S^0$, considering $0 < 1-\rho^d = \frac{\exp(W^d S^d)}{1+\exp(W^d S^d)} < 1$. The orderliness property is in favor of the *tree generation rule* by limiting the tree depth to be 3 to avoid too-small-region nodes (i.e., over-parsing). Besides, it also encourages S_R^3 to be small-region and irredundant.

3.3 Convolution with Redundancy Reduction

According to the *tree generation rule*, the nodes close to the parent node capture global layout and contain spatially redundant information, while the nodes far away from the parent node capture fine details and contain less redundant information. This rule is followed and obeyed in a data-driven manner. Based on this effective rule, we design an effective scheme to reduce the redundancy in these nodes.

According to [3, 29], the feature maps with redundant information can be downsampled into a smaller size. Motivated by these works, we downsample $S_L^1 \in \mathcal{R}^{c\times h\times w}$, $S_L^2 \in \mathcal{R}^{c\times h\times w}$, and $S_L^3 \in \mathcal{R}^{c\times h\times w}$ by a factor of 8, 4, and 2 respectively and have:

$$\tilde{S_L^1} = g_8(S_L^1),\ \tilde{S_L^2} = g_4(S_L^2),\ \tilde{S_L^3} = g_2(S_L^3), \tag{3}$$

where $g_m(\cdot)$ is a downsampling function with a shrinking factor of m. Then, the new set of downsampled nodes $\{\tilde{S}_L^1, \tilde{S}_L^2, \tilde{S}_L^3, S_R^3\}$ are fed into the vanilla convolution. Formally, let $\mathcal{W} \in \mathcal{R}^{c'\times c\times k\times k}$ denote a $k\times k$ convolution kernel where c' is the output channels. Y is the output of the convolution. We have:

$$\begin{aligned}\tilde{Y}_L^1 &= f_{\mathcal{W}_L^1}(\tilde{S}_L^1) \in \mathcal{R}^{c'\times\frac{h}{8}\times\frac{w}{8}},\ \tilde{Y}_L^2 = f_{\mathcal{W}_L^2}(\tilde{S}_L^2) \in \mathcal{R}^{c'\times\frac{h}{4}\times\frac{w}{4}},\\ \tilde{Y}_L^3 &= f_{\mathcal{W}_L^3}(\tilde{S}_L^3) \in \mathcal{R}^{c'\times\frac{h}{2}\times\frac{w}{2}},\ Y_R^3 = f_{\mathcal{W}_R^3}(S_R^3) \in \mathcal{R}^{c'\times h\times w},\end{aligned} \tag{4}$$

where $f_{\mathcal{W}}(\cdot)$ is a $k\times k$ vanilla convolution with $\mathcal{W}$ as its convolutional filter. Next, we upsample $\tilde{Y}_L^1 \in \mathcal{R}^{c'\times\frac{h}{8}\times\frac{w}{8}}$, $\tilde{Y}_L^2 \in \mathcal{R}^{c'\times\frac{h}{4}\times\frac{w}{4}}$, and $\tilde{Y}_L^3 \in \mathcal{R}^{c'\times\frac{h}{2}\times\frac{w}{2}}$ to recover their original size $c'\times h\times w$:

$$Y_L^1 = G_8(\tilde{Y}_L^1),\ Y_L^2 = G_4(\tilde{Y}_L^2),\ Y_L^3 = G_2(\tilde{Y}_L^3), \tag{5}$$

where $G_m(\cdot)$ is a upsampling function with a expansion factor of m. Finally, we obtain the convolved feature maps $\{Y_L^1, Y_L^2, Y_L^3, Y_R^3\}$. Please refer to Fig. 3 for more details.

It may be argued how the *tree generation rule* is obeyed, e.g., why the redundancy of S_L^1 is larger than that of S_R^3 in our TreeConv. Actually, although the learning process is unsupervised in our formulation (i.e., Eq. (3)), we consider the redundancy has been ordered as $S_L^1 > S_L^2 > S_L^3 > S_R^3$. Hence, we downsample S_L^1, S_L^2, and S_L^3 by a factor of 8, 4, and 2, respectively. Guided by this human knowledge (or influenced by the factors of downsampling), the network automatically encourages the redundancy of S_L^1 to be the largest and the redundancy of S_R^3 to be the smallest.

3.4 Fusing Feature Maps

Combining the convolved feature maps to form the output can be achieved by chasing the reversed path of the parsed tree from the bottom up. Specifically, we first combine (Y_L^3, Y_R^3) to form Y_R^2; then, we combine (Y_L^2, Y_R^2) to form Y_R^1; finally, we combine (Y_L^1, Y_R^1) to obtain the final output O. Mathematically, this process is equivalent to concatenate $\{Y_L^1, Y_L^2, Y_L^3, Y_R^3\}$ with respect to the channel axis *in order*, then perform a 1×1 vanilla convolution for channel fusion. Fig. 3 shows the process of node fusion.

3.5 Comparison With Previous Works

The recently proposed AOGNet [25] also attempts to combine the grammatical model with neural architectures. They design a handcrafted grammatical rule, based on which they manually design the network architecture. AOGNet has several limitations. **First**, their grammatical rule is handcrafted, fixed, and lacks theoretical explanation, which may reduce the capacity of the network. **Second**, AOGNet considers a concatenation as an AND operation and a summation as OR operation, which may be inappropriate. Mathematically, it is appropriate to consider an OR operation as a summation with some probability because an OR operation is defined as a switch to determine which path to go ahead. However, in a DNN, a summation and a concatenation share similar formulations, i.e., they are written as $w_1x + w_1y$ and $w_1x + w_2y$ respectively. Whether it is appropriate to formulate an AND node as weighted summation is open to question. **Third**, the AND-OR operations in AOGNet are performed on the channel axis but NOT spatial axis of feature maps, which may reduce the explainability and interpretability of the model. Actually, so far, finding the meaning of different channels is still an active research topic. In contrast to AOGNet, our TreeConv utilizes trees to grammatically recognize the images. Our trees are adaptive, flexible, and personalized, i.e., *our trees in different images are connected to different*

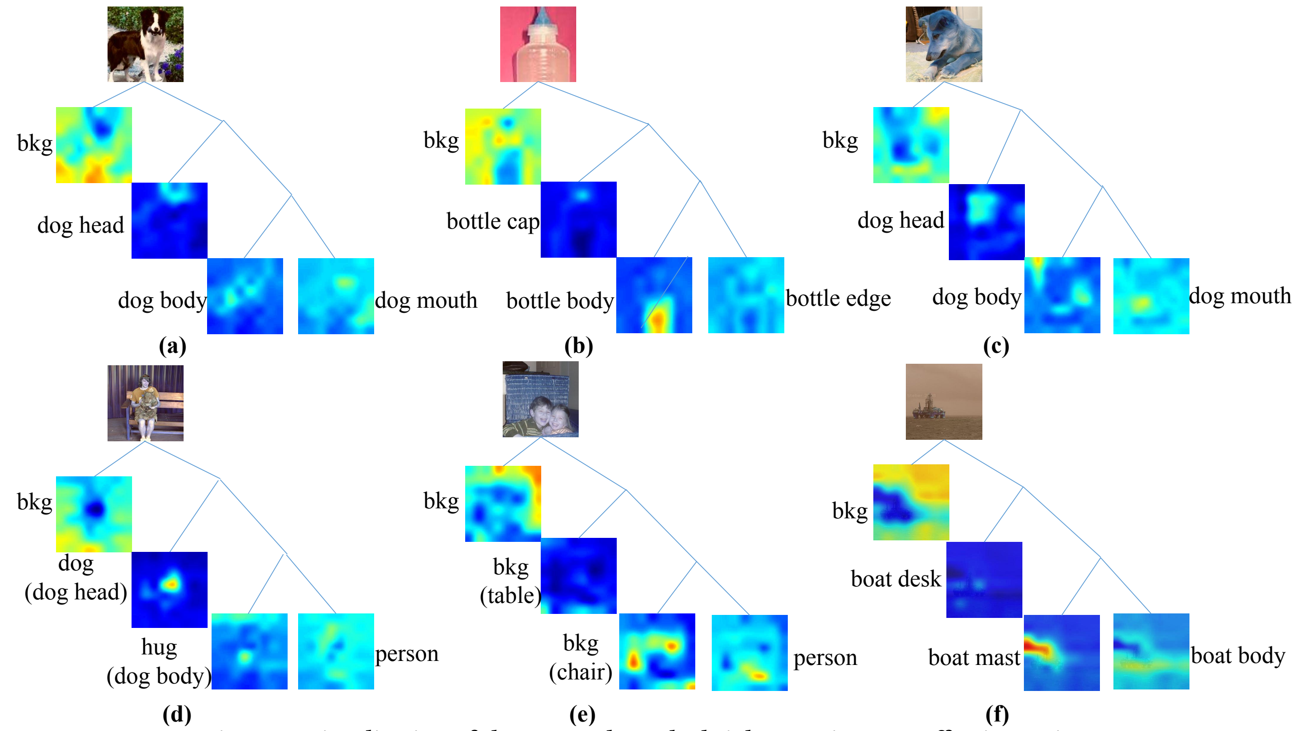

Figure 4: Visualization of the trees, where the brighter regions are effective regions.

regions adaptively (see Section 4.2). More importantly, we do not impose any predefined concept on these tree nodes. We argue that each node can hold a special concept for each class of objects after being learned on a dataset. Hence, the grammars we learned are flexible. Besides, our TreeConv is performed on the spatial axis by parsing the images into different regions according to their level of abstraction, making our TreeConv more consistent with human perception.

4 EXPERIMENTS

4.1 Explainable Visualization on ImageNet

To explore what grammars our TreeConv has learned implicitly, we first visualize the trees of the TreeConv on the ImageNet [39] classification dataset of $1k$ categories, which is one of the largest benchmarks in computer vision. The models are trained on the 1.28M training images. We select the widely used and representative ResNet50 as a baseline and follow the standard experimental protocol [16] for training.

In the conventional works on neural network explanation, interpretable visualization is challenging because the feature maps have many channels. Directly plotting the feature maps lacks interpretability. To tackle this problem, many complicated techniques have been proposed, e.g., CAM [70]. In contrast, interpretable visualization is straightforward in our TreeConv. We can visualize the trees for explanations by printing the component ratio defined in Eq. (1) with respect to the input feature maps (i.e., ρ^0, $(1-\rho^0)\rho^1$, $(1-\rho^0)(1-\rho^1)\rho^2$, and $(1-\rho^0)(1-\rho^1)(1-\rho^2)$, or *called explanation maps* for simplification.) The meaning of the explanation maps can be found in the bottom right corner of Fig. 3. In Fig. 4, we randomly sample six images from the ImageNet *val* set and visualize their explanation maps in the last convolutional layer of ResNet50.

Category-based grammars. We have observed some visual grammars implicitly learned by our TreeConv, especially category-based grammars. For example, Fig. 4 (a) and (c) are images of two dogs. **Surprisingly**, the explanation maps of these two images share the same contents, i.e., both trees are organized as $(background, (dog\ head, (dog\ body, dog\ mouth)))$. Moreover, Fig. 4 (d) also contains a dog hugged by a person. The sub-tree containing the dog in the explanation maps in Fig. 4 (d) is organized as $(background, (dog\ head, dog\ body))$, which is similar to that of the dogs in Fig. 4 (a) and (c). In contrast, the explanation maps of a bottle (Fig. 4 (b)) is quite different from that of the dogs. These comparisons verify that our TreeConv has learned some united category-based grammars for different object categories.

Personalized trees. We can find that our trees in different images focus on different regions adaptively, demonstrating that our trees are adaptive, flexible, and personalized.

Human perception. We have observed that the hierarchy of some explanation maps is consistent with human perception. The backgrounds are first parsed. Then, the informative regions featured by the objects are gradually discovered and parsed later. For example, in Fig. 4 (d), the background, the head of the dog, the body of the dog, and the person are hierarchically filtered out one by one, which is different from conventional works on neural network explanation that just filter out the background.

Discussion. In Section 3.2, the orderliness property indicates that a sparser matrix ρ corresponds to finer details and lower redundancy. We provide more explanations here. Theoretically, the wavelet theory [2] has proved that the global information (i.e., low-frequency information) is redundant in an image, while the finer details (i.e., high-frequency information) are sparse. Therefore, a sparser matrix ρ would correspond to finer details and lower redundancy. Biologically, the human perception system also considers the global dependencies as redundant and scans them quickly, and then pays attention to the sparse finer details. Experimentally, we also observe the finer details correspond to a sparser matrix ρ, as shown in Fig. 4.

4.2 ImageNet Classification

To demonstrate the effectiveness of our TreeConv, we further compare our top-1 accuracy of TreeConv with that of the most representative CNNs on the ImageNet classification to validate the effectiveness and generalization performance of our TreeConv. For a fair comparison, we examine top-1 accuracy on the 224×224 single/center-crop-single-scale images. Our baseline model is the representative ResNet50 and EfficientNet-B0 [44]. Note that, the top-1 accuracy of the baseline equals to the official results and the model zoo (Caffe; Tensorflow; Pytorch[1]). For our Tree-ResNet50, all the 3×3 convolution in the standard ResNet50 are replaced with TreeConv. The Tree-EfficientNet-B0 shares the same replacement. Moreover, we also compare our method with the recent state-of-the-art models, i.e., bL-ResNet50 [1], Oct-ResNet50 [3], AOGNet [25]. Best-performing *attention-based* methods like LRN [19], SENet [20], non-local networks (NLN [55]), and ACNet [53] are also compared.

Compare with existing models. The comparisons in terms of the top-1 validation accuracy are illustrated in Table 1 (a). As depicted, our Tree-ResNet50 surpasses all the compared methods, including state-of-the-art methods like bL-ResNet50 [1] as well as Oct-ResNet50 [3]. More importantly, our Tree-ResNet50 performs approximately **1.4%** better than the compared standard ResNet50 (77.8% vs. 76.4%) with nearly the same parameter number. This improvement is quite significant in ImageNet society.

Compare with attention mechanisms. Since both TreeConv and attention-based methods have long-range dependency modeling, we also compare our TreeConv with attention mechanisms in Table 1 (a). As shown, our Tree-ResNet50 with fewer parameters surpasses the best-performing *attention-based* methods like NLN [55], LRN [19] and SENet [20]]. This improvement verifies the advantage of our TreeConv over previous attention based methods in capturing long-range dependencies on large-scale image classification benchmark.

Parameter analysis. Consistent with discussion in Section 5, the parameter number in our Tree-ResNet50 is exactly the same as that in ResNet-50 (25.56M vs 25.57M, see Table 1 (b)). To validate the efficient learning capacity of our TreeConv, we design a lightweight version of Tree-ResNet50 by further reducing the channel number in both c and c', i.e., $c \leftarrow \frac{c}{4}, c' \leftarrow \frac{c'}{4}$. We denote this version as "Tree-ResNet50(*)". The results in Table 1 (b) illustrate that Tree-ResNet50(*) obtains limited performance drop when is compared with Tree-ResNet50 (76.2% vs 77.8%). However, the parameter number of Tree-ResNet50(*) is nearly a half of that of Tree-ResNet50. When compared with the lightweight competitors such as the recently proposed Oct-ResNet26 [3], generalACNet[53], AOGNet [25], and EfficientNet-B0 [44], our Tree-EfficientNet-B0 achieves the best-performing accuracy with significantly fewer parameters.

Table 1: Comparison on ImageNet *val* top-1 accuracies and parameter numbers. "attention?" is used to denote whether a method is attention-based. d=1: depth=1.

(a) Classification Performance.

Method	top-1	#params	attention?
ResNet50	76.4	25.56M	
MultiScale-ResNet50	76.9	25.56M	
bL-ResNet50 [1]	76.9	26.2M	
Oct-ResNet50 [3]	77.3	25.6M	
LRN [19]	77.3	23.3M	✓
NLN [55]	77.2	27.7M	✓
SENet [20]	77.3	28.1M	✓
Tree-ResNet50 (d=1)	76.7	25.56M	
Tree-ResNet50 (d=2)	77.3	25.56M	
Tree-ResNet50 (d=3)	**77.8**	**25.57M**	
Tree-ResNet50 (d=4)	77.6	25.57M	

(b) Parameter analysis.

Method	top-1	#params	attention?
R-MG-34 [22]	75.5	32.9M	
Oct-ResNet26 [3]	75.9	16.0M	
generalACNet [53]	76.2	19.80M	✓
AOGNet [25]	73.8	4.2M	
EfficientNet-B0[44]	76.3	5.28M	
Tree-ResNet50(*)	76.2	13.82M	
Tree-EfficientNet-B0	**77.8**	**5.30M**	

(c) Computational cost analysis.

Method	images/sec	Memory
ResNet50	198	8.6 GB
Tree-ResNet50	151	8.8 GB
Tree-ResNet50(*)	**223**	**8.3 GB**

Computation complexity. To validate the efficiency of our TreeConv, we compare our Tree-ResNet with the standard ResNet50 in terms of inference speed and memory usage. Considering the theoretical gain of parameter numbers might not be equal to the practical gain. We also compare with the lightweight version Tree-ResNet50(*). For a fair comparison, all methods are trained in the same desktop with 8 Titan Xp GPUs. As shown in Table 1 (c), our Tree-ResNet50 performs about 20% faster (151 vs 198 images/second/GPU) than ResNet50 with about 2% more (8.8GB vs 8.6GB) memory usage. The reason is that the size of feature maps in our Tree-ResNet50 is smaller than the vanilla convolution in ResNet50, due to the downsampling operation. Besides, our Tree-ResNet(*) has the least computational cost, verifying the effectiveness of our TreeConv.

Comparison with multi-scale learning. As Fig. 3 shows, our TreeConv implicitly employs multi-scale learning by using four feature scales (i.e., 1, 1/2, 1/4, and 1/8). To validate the superiority of our TreeConv over multi-scale learning, we replace all the 3×3 convolution in the standard ResNet50 with four parallel branches. In each branch, an $s\times$ downsampling, a 3×3 convolution, and an

[1]https://github.com/Cadene/pretrained-models.pytorch

$s\times$ upsampling are performed in order ($s \in \{1, 2, 4, 8\}$ respectively). Like our TreeConv, the output channels of each branch is $c' = \frac{c}{4}$. Finally, the outputs of the four branches are concatenated, forming the standard multi-scale learning. The result of multi-scale learning is shown in Table 1 (a), which has a 0.9% degradation compared with our TreeConv, confirming the effectiveness of our tree structure.

Analysis of different tree depth. According to the *tree generation rule*, the depth of the tree is a hyper-parameter and needs to be empirically set as a constant during the training phase. To provide more insights into the tree depth, we also conduct the experiment to compare their performance on the ImageNet classification benchmark. As shown in Table 1 (a), Tree-ResNet50 (depth=1) performs much worse than Tree-ResNet50 (depth=3) but still slightly better than the baseline ResNet50. This validates the effectiveness of our TreeConv. Tree-ResNet50 (depth=2) and Tree-ResNet50 (depth=4) perform slightly worse than Tree-ResNet50 (depth=3), which may be attributed to under-parsing and over-parsing, respectively. This also illustrates that our TreeConv is somewhat insensitive to the depth of the tree.

4.3 Analysis on CIFAR10

To demonstrate the generality of our method, we deploy TreeConv to another best-performing architecture, i.e., DARTSNet [33]. Different from ResNet50, DARTSNet is not manually designed by a human. It is automatically searched by using a neural architecture search method. Note that, DARTSNet does not consist of a vanilla convolution operation. Instead, it contains 3×3 separable convolutions and 3×3 dilated convolutions. We replace its separable convolutions and dilated convolutions with our TreeConv and retrain the model. The comparison results with recent state-of-the-art methods are presented in Table 2, showing that our Tree-DARTSNet achieves new state-of-the-art performance.

4.4 COCO 2017 Object Detection and Segmentation

We have demonstrated the effectiveness of our TreeConv in image classification; next, we evaluate it on two higher-level tasks, i.e., object detection and segmentation, which are more in need of learning visual grammar. Without loss of generality, we use one of the largest detection and segmentation benchmarks, i.e., COCO 2017 [28], to evaluate the effectiveness of our TreeConv. Following the standard protocol, we finetune the models trained on ImageNet [39] for transferring to detection & segmentation with the frozen BN parameters.

We conduct experiments on the Mask-RCNN baseline [14], which uses a ResNet50-FPN backbone. Specifically, we replace all the 3×3 layers in ResNet50 with our TreeConv layers. The models are trained in the COCO train2017 set and evaluated in the COCO val2017 set. We report the standard COCO metrics of Average Precision (AP) for bounding box detection and instance segmentation.

Table 3 shows the comparison of our TreeConv *vs.* NLN [55] *vs.* the standard CNN. Our TreeConv improves over standard CNN by 1.8% box AP and 2.2% mask AP. This may be attributed to two reasons. **First**, our implicitly learned visual grammar benefits the image recognition task. **Second**, Our TreeConv obtains non-local dependencies by learning a grammar that represents the whole image. We have also found NLN is 0.8% box AP worse than our TreeConv. This suggests that although NLN is also suitable for global inference, its capacity is weaker than our TreeConv due to inflexible-globalization, i.e., it imposes the same global information to every pixel of a feature map. This phenomenon has also been observed experimentally by [74] and theoretically discussed in [34]. In contrast, our TreeConv achieves flexible-globalization by learning flexible visual grammars.

Table 2: CIFAR10 Analysis.

Method	error	#params
DenseNet[21]	3.46	25.6M
NASNet-A[75]	2.83	3.1M
AmoebaNet-A[37]	3.12	3.1M
PNAS[31]	3.41	3.2M
ENAS[36]	2.89	4.6M
DARTS[33]	2.76	3.3M
Tree-DARTS	**2.68**	**3.3M**

Table 3: Detection and segmentation on COCO 2017 with Mask-RCNN. "attention?": attention-based or not.

Method	AP^{bbox}	AP^{mask}	attention?
Mask-RCNN(CNN)	38.0	34.6	
Mask-RCNN(NLN)	39.0	35.5	✓
Mask-RCNN(TreeConv)	**39.8**	**36.8**	

Table 4: Comparison on CUHK03 Person ReID.

Method	Rank-1
IDE+DaF [61]	26.4
IDE+XQ.+Re-ranking [68]	34.7
DPFL [4]	40.7
SVDNet [43]	41.5
TriNet + Era. [69]	55.5
TriNet + Era.(Our reproduction)	62.0
TriNet + Era. + Tree	**66.2**
TriNet + Era. + reranking	61.2
TriNet + Era. + reranking + Tree	**67.4**
MGN [54]	68.0
MGN + Tree	**71.4**

4.5 CUHK03 Person Re-identification (ReID)

In a general sense, all the above tasks (image classification, object detection, and segmentation) belong to visual classification since detection and segmentation can be seen as categorizing regions and pixels for a given image. The effectiveness of our TreeConv beyond visual classification remains uninvestigated. In the following, we investigate a completely different task, i.e., ReID, which is fundamental in video surveillance for keeping the safety of society [49–52]. ReID refers to the problem of re-identifying individuals across cameras. Mathematically, ReID is a matching problem rather than a classification problem, because it requires to calculate distance metric between two given images. As is proved by [57], learning image grammar benefits the distance metric learning, therefore validating the effectiveness of our TreeConv in ReID can verify the contribution of the visual grammar learned by our TreeConv.

Dataset & Metric. We conduct experiments on the CUHK03 dataset [24], which is one of the largest databases for ReID. This database contains 14,096 images of 1,467 pedestrians. Each person is observed by two disjoint camera views and is shown in 4.8 images on average in each view. We follow the **new** standard setting of using CUHK03 [69], where 767 individuals are regarded as the training set, and another 700 individuals are considered as the testing set without sharing the same individuals. For the evaluation, the testing set is further divided into a gallery set of images and a query set. We use the standard rank-1 as the evaluation metric. The standard protocol [69] is employed for the training.

Result Analysis. In Table 4, we compare with the current representative state-of-the-art models, i.e., BOW+XQDA [66], PUL [7], LOMO+XQDA [26], IDE [67], IDE+DaF [61], IDE+XQ.+Re-ranking [68], DPFL [4], and the newly proposed methods SVDNet [43], TriNet + Era. [69], TriNet + Era. + Reranking [69], and the current best-performing MGN [54]. Our baseline is [69] and MGN [54], which use ResNet50 as the feature extractors. All the 3×3 convolutions in ResNet50 are replaced with our TreeConv, forming our models TriNet + Era. + Tree, TriNet + Era. + reranking + Tree, and MGN + Tree. The results in Table 4 show that our models achieve a new state-of-the-art performance, i.e., a rank-1 accuracy of 71.4%. We can also observe that TreeConv surpasses its baseline by a clear margin (71.4% *vs.* 68.0% for MGN + Tree *vs.* MGN). The consistent performance gains are also obtained on improving both the TriNet + Era and TriNet + Era. + reranking methods. This verifies the effectiveness of TreeConv on visual matching tasks like ReID.

5 CONCLUSION

This work presented a novel tree convolution (TreeConv) operation to improve deep neural networks. Through explicitly modeling the hierarchy of internal feature maps and implicitly learning the visual grammars among them, our TreeConv not only improves the network capability and reduces the network redundancy naturally, but also provides explainable visualization results on ImageNet. However, this work only focuses on implicit visual grammar learning. Can we learn the explicit visual grammar? What is the explicit visual grammar? These questions are so difficult to answer to date. Future works will focus on these directions.

ACKNOWLEDGMENTS

This work was supported in part by the State Key Development Program under Grant 2018YFC0830103, in part by DARPA XAI N66001-17-2-4029, in part by the National Natural Science Foundation of China under Grant U1811463, 61622214, and 61836012, and in part by Natural Science Foundation of Guangdong Province of No. 2019A1515010939 and 2017A030312006.

REFERENCES

[1] Chun-Fu (Richard) Chen, Quanfu Fan, Neil Mallinar, Tom Sercu, and Rogério Schmidt Feris. 2019. Big-Little Net: An Efficient Multi-Scale Feature Representation for Visual and Speech Recognition. In *Proceedings of International Conference on Learning Representations (ICLR).*

[2] Tianshui Chen, Liang Lin, Wangmeng Zuo, Xiaonan Luo, and Lei Zhang. 2018. Learning a wavelet-like auto-encoder to accelerate deep neural networks. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI).*

[3] Yunpeng Chen, Haoqi Fan, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. 2019. Drop an Octave: Reducing Spatial Redundancy in Convolutional Neural Networks with Octave Convolution. *Proceedings of International Conference on Computer Vision (ICCV)* (2019).

[4] Yanbei Chen, Xiatian Zhu, and Shaogang Gong. 2017. Person Re-Identification by Deep Learning Multi-Scale Representations. In *Proceedings of Computer Vision and Pattern Recognition (CVPR).* 2590–2600.

[5] François Chollet. 2017. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proceedings of Computer Vision and Pattern Recognition (CVPR).* 1800–1807.

[6] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. 2017. Deformable Convolutional Networks. In *Proceedings of International Conference on Computer Vision (ICCV).*

[7] Hehe Fan, Liang Zheng, Chenggang Yan, and Yi Yang. 2018. Unsupervised Person Re-identification: Clustering and Fine-tuning. *TOMM* 14, 4 (2018), 83:1–83:18.

[8] Hongyang Gao and Shuiwang Ji. 2019. Graph Representation Learning via Hard and Channel-Wise Attention Networks. In *Proceedings of International Conference on Knowledge Discovery & Data Mining (KDD).* 741–749.

[9] G. Gazdar. 1985. *Generalized phrase structure grammar.* Harvard University Press.

[10] Dileep George, Wolfgang Lehrach, Ken Kansky, Miguel Lázaro-Gredilla, Christopher Laan, Bhaskara Marthi, Xinghua Lou, Zhaoshi Meng, Yi Liu, Huayan Wang, et al. 2017. A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs. *Science* 358, 6368 (2017), eaag2612.

[11] D. George, W. Lehrach, K. Kansky, M. Lazaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang, A. Lavin, and D. S. Phoenix. 2017. A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science* (2017).

[12] F. Han and S.-C. Zhu. 2009. Bottom-up/top-down image parsing with attribute grammar. *IEEE transactions on pattern analysis and machine intelligence* 31, 1 (2009), 59–73.

[13] D. G. Hays. 1964. Dependency theory: A formalism and some observations. *Language* 40, 4 (1964), 511–525.

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of International Conference on Computer Vision (ICCV).* IEEE, 2980–2988.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of Computer Vision and Pattern Recognition (CVPR).*

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of Computer Vision and Pattern Recognition (CVPR).* 770–778.

[17] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[18] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR* abs/1704.04861 (2017).

[19] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. 2019. Local Relation Networks for Image Recognition. *Proceedings of International Conference on Computer Vision (ICCV)* (2019).

[20] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-Excitation Networks. In *Proceedings of Computer Vision and Pattern Recognition (CVPR).* 7132–7141.

[21] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2017. Densely connected convolutional networks. In *Proceedings of Computer Vision and Pattern Recognition (CVPR).*

[22] Tsung-Wei Ke, Michael Maire, and Stella X Yu. 2017. Multigrid neural architectures. In *Proceedings of Computer Vision and Pattern Recognition (CVPR).* 6665–6673.

[23] Pan Li, Zhen Qin, Xuanhui Wang, and Donald Metzler. 2019. Combining Decision Trees and Neural Networks for Learning-to-Rank in Personal Search. In *Proceedings of International Conference on Knowledge Discovery & Data Mining (KDD).* 2032–2040.

[24] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. 2014. Deepreid: Deep filter pairing neural network for person re-identification. In *Proceedings of Computer Vision and Pattern Recognition (CVPR).* 152–159.

[25] Xilai Li, Xi Song, and Tianfu Wu. 2019. AOGNets: Compositional Grammatical Architectures for Deep Learning. In *Proceedings of Computer Vision and Pattern Recognition (CVPR).* 6220–6230.

[26] Shengcai Liao, Yang Hu, Xiangyu Zhu, and Stan Z Li. 2015. Person re-identification by local maximal occurrence representation and metric learning. In *Proceedings of Computer Vision and Pattern Recognition (CVPR).* 2197–2206.

[27] Liang Lin, Guangrun Wang, Rui Zhang, Ruimao Zhang, Xiaodan Liang, and Wangmeng Zuo. 2016. Deep structured scene parsing by learning with image descriptions. In *Proceedings of Computer Vision and Pattern Recognition (CVPR).* 2276–2284.

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Proceedings of European Conference on Computer Vision*

(ECCV). Springer, 740–755.

[29] Tony Lindeberg. 2013. *Scale-space theory in computer vision*. Vol. 256. Springer Science & Business Media.

[30] Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics* 4 (2016), 521–535.

[31] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. 2018. Progressive neural architecture search. In *Proceedings of European Conference on Computer Vision (ECCV)*. 19–34.

[32] Donghua Liu, Jing Li, Bo Du, Jun Chang, and Rong Gao. 2019. DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation. In *Proceedings of International Conference on Knowledge Discovery & Data Mining (KDD)*. 344–352.

[33] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable Architecture Search. In *Proceedings of International Conference on Learning Representations (ICLR)*.

[34] David Lopez-Paz, Robert Nishihara, Soumith Chintala, Bernhard Scholkopf, and Léon Bottou. 2017. Discovering causal signals in images. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 6979–6987.

[35] Lili Mou, Ge Li, Lu Zhang, Tao Wang, and Zhi Jin. 2016. Convolutional Neural Networks over Tree Structures for Programming Language Processing. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI)*. 1287–1293.

[36] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. 2018. Efficient Neural Architecture Search via Parameter Sharing. In *Proceedings of International Conference on Machine Learning (ICML)*. 4092–4101.

[37] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. 2019. Regularized Evolution for Image Classifier Architecture Search. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI)*. 4780–4789.

[38] Deboleena Roy, Priyadarshini Panda, and Kaushik Roy. 2020. Tree-CNN: A hierarchical Deep Convolutional Neural Network for incremental learning. *Neural Networks* 121 (2020), 148–160.

[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.

[40] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 4510–4520.

[41] Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron C. Courville. 2018. Neural Language Modeling by Jointly Learning Syntax and Lexicon. In *Proceedings of International Conference on Learning Representations (ICLR)*.

[42] Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered Neurons: Integrating Tree Structures into Recurrent Neural Networks. *Proceedings of International Conference on Learning Representations (ICLR)* (2019).

[43] Yifan Sun, Liang Zheng, Weijian Deng, and Shengjin Wang. 2017. SVDNet for Pedestrian Retrieval. In *Proceedings of International Conference on Computer Vision (ICCV)*. 3820–3828.

[44] Mingxing Tan and Quoc V. Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of International Conference on Machine Learning (ICML)*.

[45] Wei Tang, Pei Yu, and Ying Wu. 2018. Deeply learned compositional models for human pose estimation. In *Proceedings of European Conference on Computer Vision (ECCV)*. 190–206.

[46] Wei Tang, Pei Yu, Jiahuan Zhou, and Ying Wu. 2017. Towards a unified compositional model for visual pattern modeling. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 2784–2793.

[47] Xiaoli Tang, Tengyun Wang, Haizhi Yang, and Hengjie Song. 2019. AKUPM: Attention-Enhanced Knowledge-Aware User Preference Model for Recommendation. In *Proceedings of International Conference on Knowledge Discovery & Data Mining (KDD)*. 1891–1899.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems (NeurIPS)*. 5998–6008.

[49] Guangcong Wang, Jianhuang Lai, Peigen Huang, and Xiaohua Xie. 2019. Spatial-Temporal Person Re-Identification. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. 8933–8940. https://doi.org/10.1609/aaai.v33i01.33018933

[50] Guangcong Wang, Jianhuang Lai, and Xiaohua Xie. 2018. P2SNet: Can an Image Match a Video for Person Re-Identification in an End-to-End Way? *IEEE Trans. Circuits Syst. Video Techn.* 28, 10 (2018), 2777–2787.

[51] Guangcong Wang, Jian-Huang Lai, Wenqi Liang, and Guangrun Wang. 2020. Smoothing Adversarial Domain Attack and P-Memory Reconsolidation for Cross-Domain Person Re-Identification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[52] Guangrun Wang, Guangcong Wang, Xujie Zhang, Jianhuang Lai, Zhengtao Yu, and Liang Lin. 2020. Weakly Supervised Person Re-ID: Differentiable Graphical Learning and A New Benchmark. In *IEEE Transactions on Neural Networks and Learning Systems (T-NNLS)*.

[53] Guangrun Wang, Keze Wang, and Liang Lin. 2019. Adaptively Connected Neural Networks. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 1781–1790.

[54] Guanshuo Wang, Yufeng Yuan, Xiong Chen, Jiwei Li, and Xi Zhou. 2018. Learning Discriminative Features with Multiple Granularities for Person Re-Identification. In *Proceedings of ACM Multimedia Conference on Multimedia Conference (ACM MM)*. 274–282.

[55] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. 2018. Non-local neural networks. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 7794–7803.

[56] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of International Conference on Knowledge Discovery & Data Mining (KDD)*. 950–958.

[57] Yuanlu Xu, Liang Lin, Wei-Shi Zheng, and Xiaobai Liu. 2013. Human re-identification by matching compositional template with cluster sampling. In *Proceedings of International Conference on Computer Vision (ICCV)*. 3152–3159.

[58] Y. Xu, X. Liu, Y. Liu, and S.-C. Zhu. 2016. Multi-view people tracking via hierarchical trajectory composition. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*.

[59] Dani Yogatama, Yishu Miao, Gabor Melis, Wang Ling, Adhiguna Kuncoro, Chris Dyer, and Phil Blunsom. 2018. Memory architectures in recurrent neural network language models. In *Proceedings of International Conference on Learning Representations (ICLR)*.

[60] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. 2019. Free-Form Image Inpainting With Gated Convolution. In *Proceedings of International Conference on Computer Vision (ICCV)*. 4470–4479. https://doi.org/10.1109/ICCV.2019.00457

[61] Rui Yu, Zhichao Zhou, Song Bai, and Xiang Bai. 2017. Divide and Fuse: A Re-ranking Approach for Person Re-identification. In *Proceedings of British Machine Vision Conference (BMVC)*.

[62] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. 2019. Self-Attention Generative Adversarial Networks. In *Proceedings of International Conference on Machine Learning (ICML)*. 7354–7363.

[63] Ruimao Zhang, Liang Lin, Guangrun Wang, Meng Wang, and Wangmeng Zuo. 2019. Hierarchical scene parsing by weakly supervised learning with image descriptions. *IEEE transactions on pattern analysis and machine intelligence* 41, 3 (2019), 596–610.

[64] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 6848–6856.

[65] Y. Zhao and S.-C. Zhu. 2011. Image parsing with stochastic scene grammar. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[66] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. 2015. Scalable person re-identification: A benchmark. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[67] Liang Zheng, Yi Yang, and Alexander G Hauptmann. 2016. Person re-identification: Past, present and future. *arXiv preprint arXiv:1610.02984* (2016).

[68] Zhun Zhong, Liang Zheng, Donglin Cao, and Shaozi Li. 2017. Re-ranking Person Re-identification with k-Reciprocal Encoding. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 3652–3661.

[69] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2017. Random Erasing Data Augmentation. *arXiv preprint arXiv:1708.04896* (2017).

[70] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning Deep Features for Discriminative Localization. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 2921–2929.

[71] Yichao Zhou, Shaunak Mishra, Jelena Gligorijevic, Tarun Bhatia, and Narayan Bhamidipati. 2019. Understanding Consumer Journey using Attention based Recurrent Neural Networks. In *Proceedings of International Conference on Knowledge Discovery & Data Mining (KDD)*. 3102–3111.

[72] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning Tree-based Deep Model for Recommender Systems. In *Proceedings of International Conference on Knowledge Discovery & Data Mining (KDD)*. 1079–1088.

[73] Song-Chun Zhu and David Mumford. 2007. *A Stochastic Grammar of Images*. Now Publishers Inc., Hanover, MA, USA.

[74] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. 2019. Deformable ConvNets V2: More Deformable, Better Results. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 9308–9316.

[75] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 8697–8710.

REPRODUCIBILITY & IMPLEMENTATION DETAILS

In the following, we present more details of our TreeConv for third-party reproductivity.

Number of channels. For fair comparison in terms of both *parameter number* and *tensor size*, the output tensors c' in Eq. (4) is set as $c' = \frac{c}{4}$. Therefore, the parameter number in our TreeConv is exactly the same as that in vanilla convolution. Note that the size of feature maps in our TreeConv is smaller than that in vanilla convolution, due to the downsampling operation in our model. It is worth mentioning that due to the efficient learning capacity of our tree convolution, further reducing the channel number in both c and c' (e.g., $c \leftarrow \frac{c}{4}, c' \leftarrow \frac{c'}{4}$) does not affect the performance of our model too much, while greatly reducing the computational cost (see experiments in Section 4.2).

Downsampling and upsampling. As is discussed in Section 3.3, the feature maps with redundant information can be downsampled into a smaller size to reduce computational cost. For downsampling, we utilize average pooling (2, 2), (4, 4), and (8, 8), respectively, where (k, s) means that the pooling kernel and the stride are k×k and s, respectively. For upsampling, we have tried two kinds of strategies, i.e., nearest interpolation and bilinear interpolation. According to our experiments, these two interpolations have similar top-1 accuracies. Therefore, in our experiments, we simply pick up the nearest interpolation.

Variants of vanilla convolutions. Our TreeConv can also be adapted to other popular variants of vanilla convolutions such as depth-wise, dilated, group convolution, and so on because our TreeConv does not directly change the convolutional filter. This can be done by simply replacing the vanilla convolution $f_{\mathcal{W}}(\cdot)$ in Eq. (4) with other variants of convolutions (see experiments in Section 4.3).

Deploying TreeConv into backbone networks. TreeConv is compatible with vanilla convolution and can be inserted into state-of-the-art backbone architectures such as ResNets [15] without special adjustment. For example, we replace all of the 3×3 convolutions in ResNet50, forming the Tree-ResNet50. Similarly, we have also equipped one of the best-performing architectures, i.e., DARTSNet, in CIFAR10, forming Tree-DARTSNet (see experiments in Section 4.3).