Learning Deep Compositional Grammatical Architectures for Visual Recognition

Xilai Li[†], Tianfu Wu[†,‡,*], and Xi Song
Department of ECE[†] and the Visual Narrative Initiative[‡], North Carolina State University
{xli47, tianfu_wu}@ncsu.edu, xsong.lhi@gmail.com

Abstract

Neural architectures are the foundation for improving performance of deep neural networks (DNNs). This paper presents deep compositional grammatical architectures which harness the best of two worlds: grammar models and DNNs. The proposed architectures integrate compositionality and reconfigurability of the former and the capability of learning rich features of the latter in a principled way. They also show the platform-agnostic capability in deployment (e.g., cloud vs mobile). We utilize AND-OR Grammars (AOG) [41, 60, 59] in this paper and call the resulting networks ***AOGNets****. An AOGNet consists of a number of stages each of which is composed of a number of AOG building blocks. An AOG building block splits its input feature map into N groups along feature channels and then treat it as a sentence of N words. It then jointly realizes a phrase structure grammar and a dependency grammar in bottom-up parsing the "sentence" for better feature exploration and exploitation. It provides a unified framework for the split-transform-aggregate heuristic widely used in neural architecture design. In experiments, AOGNets are tested on three highly competitive image classification benchmarks: CIFAR-10, CIFAR-100 and ImageNet-1K. AOGNets obtain better performance than ResNets [15] and most of its variants, ResNeXts [52], DenseNets [21] and DualPathNets [3] when model sizes are comparable. AOGNets are also tested in object detection on the PASCAL VOC 2007 and 2012 [5] using the vanilla Faster R-CNN [37] system. AOGNets use smaller models and obtain better performance by about 3% mean average precision than the ResNets backbone.*

1. Introduction

1.1. Motivation and Objective

Recently, deep neural networks (DNNs) [30, 25] improved prediction accuracy significantly in many vision tasks, and even obtained superhuman performance in image classification tasks [15, 44, 21, 3]. Much of these progress have been achieved mainly through engineering network architectures which can enjoy increasing representational power (by going either deeper or wider) without sacrificing the feasibility of optimization using back-propagation with stochastic gradient descent (i.e., handling the vanishing and/or exploding gradient problem). The dramatic success does not necessarily speak to its sufficiency given the lack of theoretical underpinnings of deep neural networks at present [1]. Different methodologies are worth exploring to enlarge the scope of neural architectures for seeking better DNNs. For example, Hinton recently pointed out a crucial drawback of current convolutional neural networks: according to recent neuroscientific research, these artificial networks do not contain enough levels of structure [17, 39].

As illustrated in Fig. 1 (a), neural architecture design and search can be posed as a combinatorial search problem in a product space comprising two sub-spaces:

- The structure space which consists of all directed acyclic graphs (DAGs) with the start node representing input raw data and the end node representing task loss functions. DAGs are entailed for feasible computation.
- The node operation space which consists of all possible transformation functions for implementing nodes in a DAG, such as Convolution+BatchNorm [23]+ReLU [25] and its bottleneck implementation [15].

The structure space is almost unbounded, and the node operation space for a given structure is also combinatorial. Neural architecture design and search is an NP-hard problem due to the exponentially large space and the highly non-convex non-linear objective function to be optimized in the search. As illustrated in Fig. 1 (b), to mitigate the difficulty, neural architecture design and search have been simplified to design or search a building block structure, and then stack the same build block structure into a predefined number of stages. Fig. 1 (c) shows some examples of popular building blocks. In those hand-crafted building blocks, the best practices of finding a good building block adopt the so-called *split-transform-aggregate* heuristic. The heuristic is motivated by the well-known Hebbian principle in neuroscience, i.e., neurons fire together, then wire tighter. Put

[*]T. Wu is the corresponding author.

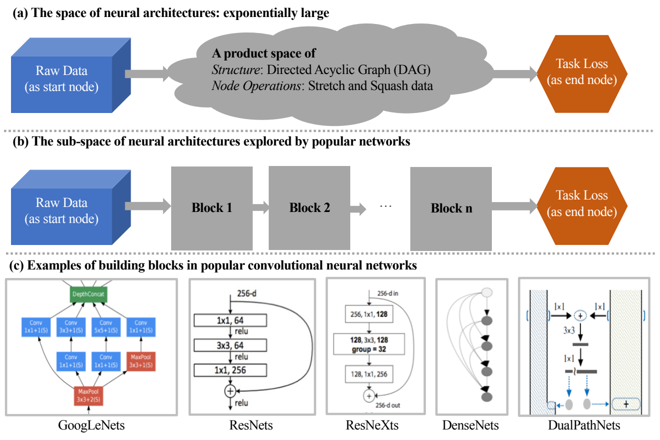
(a) The space of neural architectures: exponentially large
Raw Data (as start node)
A product space of
Structure: Directed Acyclic Graph (DAG)
Node Operations: Stretch and Squash data
Task Loss (as end node)
(b) The sub-space of neural architectures explored by popular networks
Raw Data (as start node)
Block 1
Block 2
Block n
Task Loss (as end node)
(c) Examples of building blocks in popular convolutional neural networks
GoogLeNets
ResNets
ResNeXts
DenseNets
DualPathNets

Figure 1. (a) Illustration of the space of neural architectures, (b) the building block based approaches to explored the space, and (c) examples of popular building blocks in GoogLeNets [44], ResNets [15], ResNeXts [52], DenseNets [21] and DualPathNets [3]. See text for details. (Best viewed in color)

in another word, the wisdom in designing better deep network architectures usually lies in finding network topologies which can support flexible information flows for both exploring new features and exploiting existing features in previous layers. More specifically, we observed the advantages of popular networks are:

- InceptionNets or GoogLeNets [44] embodies the split-transform-aggregate heuristic in a shallow feed-forward way. However, the filter numbers and sizes are tailored for each individual transformation, and the modules are customized stage-by-stage. Interleaved group convolutions [56] share the similar spirit, but use simpler scheme.
- ResNets [15] provide a simple yet effective solution that enables networks to enjoy going either deeper or wider without sacrificing the feasibility of optimization using back-propagation with stochastic gradient descent (i.e., handling the vanishing and/or exploding gradient problems). From the perspective of representation learning, skip-connections within a ResNet [15] contributes to effective features exploitation/reuse. They do not realize the split component.
- ResNeXts [52] add the spit component in ResNets and address the drawbacks of the Inception modules using group convolutions in the transformation.
- Deep pyramid ResNets [13] extend ResNets, which concentrate on the feature map dimension by increasing it gradually instead of by increasing it sharply at each residual unit with down-sampling.
- DenseNets [21] explicitly differentiate between information that is added to the network (i.e., exploration via split-transform) and information that is preserved (i.e., exploitation via aggregation, especially residual connections). From the perspective of representation learning, dense connection with feature maps being concatenated together in DenseNets [21] leads to effective feature exploration.
- Dual Path Networks (DPN) [3] utilize ResNet blocks and DenseNet blocks in parallel to balance feature exploitation and feature exploration.
- Deep layer aggregation networks (DLA) [54] iteratively and hierarchically merge the feature hierarchy when stacking the building blocks such as the ResNet ones.

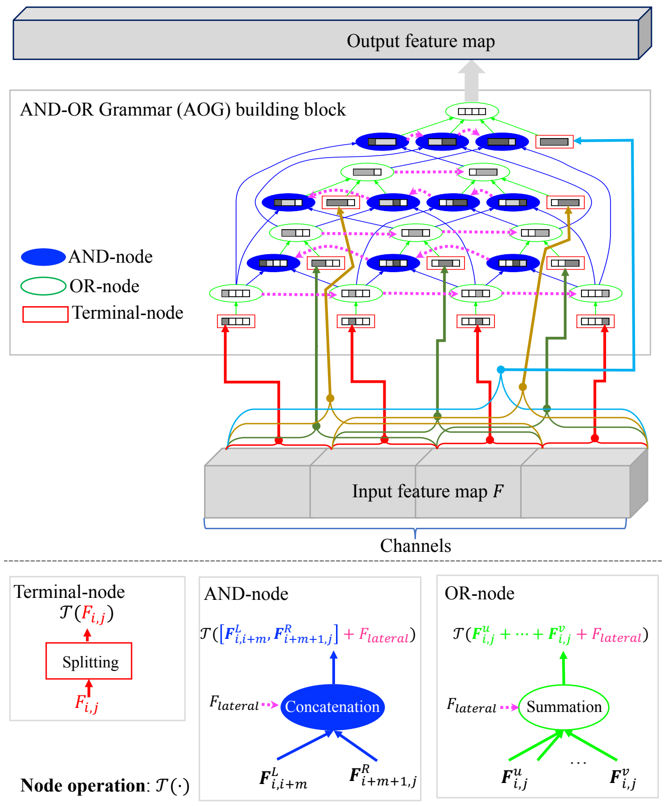

Figure 2. Illustration of the proposed AOG building block. See text for details. (Best viewed in color)

On the one hand, this paper is motivated by the technical question: Can we unify all the best practices developed in the popular networks? On the other hand, *Compositionality, reconfigurability and lateral connectivity* are well-known principles in cognitive science, neuroscience and pattern theory [9, 35, 12, 10, 26, 10]. They are fundamental for the remarkable capabilities possessed by humans, of learning rich knowledge and adapting to different environments quickly, especially in vision and language. They have not been, however, fully and explicitly integrated in DNNs. This paper aims at addressing these issues. As illustrated in Fig. 2, *this paper presents a method of deeply integrating hierarchical and compositional grammars and DNNs for harnessing the best of both worlds in representation learning.* As David Mumford pointed out, "Gram-

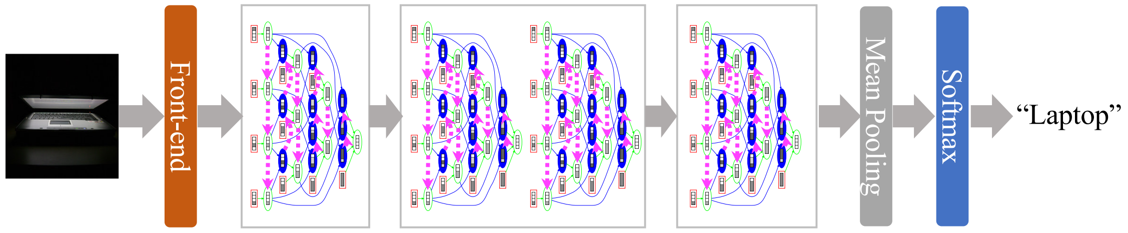

Figure 3. Illustration of an AOGNet which has 3 stages, 1 AOG building bock in the first and third stage, and 2 AOG building blocks in the second stage. Note that different stages can use different AND-OR graphs. We show the same one for simplicity. The front-end can be either a vanilla convolution or convolution+MaxPooling. (Best viewed in color)

mar in language is merely a recent extension of much older grammars that are built into the brains of all intelligent animals to analyze sensory input, to structure their actions and even formulate their thoughts.” [34].

Grammar models are well known in both natural language processing and computer vision. Image grammar [60, 6, 59, 9] was one of the dominant methods in computer vision before the recent resurgence in popularity of deep neural networks. With the recent resurgence, one fundamental puzzle arises that grammar models with more explicitly compositional structures and better analytic and theoretical potential, often perform worse than their neural network counterparts. The proposed method bridges the performance gap, which is motivated by, and aims at showing, the advantage of two nice properties of grammars which are desirable in network engineering: (i) The flexibility and simplicity of constructing different types of structural topologies based on a dictionary of primitives and a set of production rules in a principled way; and (ii) The highly expressive power and the parsimonious compactness of their explicitly hierarchical and compositional structures.

1.2. Method Overview

In this paper, we utilize AND-OR Grammars (AOG) in implementing our deep compositional grammatical architectures. We follow the general AOG framework [60, 59] and specifically extend the method presented in [41, 51] in constructing AOG building blocks. Fig. 2 shows the proposed AOG building block. We call the resulting networks **AOGNets**. An AOGNet consists of a stack of AOG building blocks. Fig. 3 illustrates a 3-stage AOGNet.

An AOG building block splits its input feature map into N groups along feature channels, and treat it as a sentence of N words. It then jointly realizes a phrase structure grammar (vertical composition) [8, 9, 7, 60, 59, 41] and a dependency grammar (horizontal connections in pink in Fig. 2) [14, 60, 10] in bottom-up parsing the “sentence” for better feature exploration and exploitation, thus embodying Mumford’s vision on grammars to analyze sensory inputs [34] in visual recognition and Hinton’s quest on more diverse neural architectures.

- The phrase structure grammar component is a 1-D special case of the method presented in [41, 51]. It can also be understood as a modified version of the well-known Cocke-Younger-Kasami (CYK) parsing algorithm in natural language processing according to a binary composition rule.
- The dependency grammar component is integrated to capture lateral connections and improve the representational flexibility and power.

In an AOG building block, each node applies some basic operation $\mathcal{T}(\cdot)$ (e.g., Conv-BatchNorm-ReLU) to its input. There are three types of nodes:

- A *Terminal-node* takes as input a channel-wise slice of the input feature map (i.e., a k-gram).
- An *AND-node* explores composition, whose input is computed by concatenating features of its syntactic child nodes, and adding the lateral connection if had;
- An *OR-node* represents alternative ways of composition in the spirit of exploitation, whose input is the element-wise sum of features of its syntactic child nodes and the lateral connection if had.

Our AOG building blocks unify the best practices developed in popular networks stated above in that,

- Terminal-nodes implement the split-transform heuristic (or group convolutions) as done in GoogLeNets [44] and ResNeXts [52], but at multiple levels (including overlapped group convolutions). They also implement the skip-connection at multiple levels. Unlike the cascade-based stacking scheme in ResNets, DenseNets and DPNs, Termninal-nodes can be computed in parallel to improve efficiency. Non-terminal nodes implement aggregation.
- AND-nodes implement DenseNet-like aggregation (i.e., concatenation) [21] for feature exploration.
- OR-nodes implement ResNet-like aggregation (i.e., summation) [15] for feature exploitation.
- The hierarchy facilitates gradual increase of feature channels as in Deep Pyramid ResNets [13], and also leads to good balance between depth and width of networks.

- The compositional structure provides much more flexible information flows than DPN [3] and the DLA [54].
- The lateral connections increase the depth of nodes on the flow without introducing extra parameters.

In experiments, we test our AOGNets on three highly competitive and widely used image classification benchmarks: the CIFAR-10 dataset and the CIFAR-100 dataset [24], and the ImageNet-1K dataset [38]. Our AOGNets obtain better performance consistently than ResNets [15] and most variants, ResNeXts [52], DenseNets [21] and DualPathNets [3] when model sizes are comparable. We also test AOGNets in object detection on the PASCAL VOC 2007 and 2012 [5]. We adopt the vanilla Faster R-CNN [37] system using AOGNets as backbone. We obtain better performance by about 3% mean average precision than the one with larger ResNets as the backbones.

2. Related Work and Our Contributions

Neural architectures are the foundation for improving performance of DNNs. The majority of existing methods are still based on hand-crafted architectures. A promising trend is to automatically learn better architectures with the long-term objective to have theoretical guarantee. So far, hand-crafted ones have better overall performance, especially on large-scale datasets such as the ImageNet benchmark [38]. We focus on hand-crafted architectures in this section. Related work on automatic neural architecture search is referred to the nice survey papers [4, 53].

Hand-crafted network architectures. After more than 20 years since the seminal work 5-layer LeNet5 [30] was proposed, the recent resurgence in popularity of neural networks was triggered by the 8-layer AlexNet [25] with breakthrough performance on ImageNet [38] in 2012. Since then, a lot of efforts were devoted to learn deeper AlexNet-like networks with the intuition that deeper is better. The VGG Net [2] proposed a 19-layer network with insights on using multiple successive layers of small filters (e.g., 3×3) A special case, 1×1 convolution, was proposed in the network-in-network [32] for reducing or expanding feature dimensionality between consecutive layers, and have been widely used in many networks. The 22-layer GoogLeNet [45] introduced the first inception module and a bottleneck scheme implemented with 1×1 convolution for reducing computational cost. The main obstacle of going deeper lies in the gradient vanishing issue in optimization, which is addressed with a new structural design, short-path or skip-connection, proposed in the Highway network [42] and popularized by the ResNets [15], especially when combined with the batch normalization [23]. More than 100 layers are popular design in the recent literature [15, 44], as well as even more than 1000 layers trained on large scale datasets such as ImageNet [22, 57]. The Fractal Net [29] and deeply fused networks [49] provided an alternative way of implementing short path for training ultra-deep networks without residuals. Complementary to going deeper, width matters in ResNets and inception based networks too [55, 52, 56]. Going beyond the first-order skip-connections in ResNets, DenseNets [21] proposed a densely connected network architecture with concatenation scheme for feature reuse and exploration, and DPNs [3] proposed to combine residuals and densely connections in an alternating way for more effective feature exploration and exploitation. DLA networks [54] further develop iterative and hierarchical aggregation schema with very good performance obtained. Most work focused on boosting spatial encoding and utilizing spatial dimensionality reduction. The squeeze-and-excitation module [19] is a recently proposed simple yet effective method focusing on channel-wise encoding. The Hourglass network [36] proposed a hourglass module consisting of both subsampling and upsampling to enjoy repeated bottom-up/top-down feature exploration.

Our AOGNet is created by intuitively simple yet principled grammars. It shares some spirit with the inception module [44], the deeply fused nets [49] and the DLA [54].

Grammar-like structures. A general framework of image grammar was proposed in [60]. Object detection grammar was the dominant approaches for object detection [6, 59, 41, 50, 31], and has recently been integrated with DNNs [46, 47]. Probabilistic program induction [43, 27, 28] has been used successfully in many settings, but has not shown good performance in difficult visual understanding tasks such as large-scale image classification and object detection. More recently, recursive cortical networks [10] have been proposed with much more data efficiency in learning which adopts the AND-OR grammar framework [60], showing great potential of grammar like structures in developing general AI systems.

Our contributions. This paper makes two main contributions in the field of deep representation learning:

- It presents a simple yet effective method of deeply integrating grammar models and DNNs, which facilitates both feature exploration and exploitation in a hierarchical and compositional way with nice balance between depth and width. In implementation, we adopt the AND-OR grammars (AOG) [41, 60, 59]. To the best of our knowledge, it is the first work that utilizes grammar models in network engineering.
- It obtains better performance than state-of-the-art networks including ResNets [15], ResNeXts [52], DenseNets [21] and DualPathNets [3] in image classification, and also achieves better object detection performance than the ResNet backbone in Faster R-CNN [37] detection systems.

3. AOGNets

In this section, we first present details of constructing the structure of our AOGNets. Then, we define node operation functions for nodes in an AOGNet. We also propose a method of simplifying the full structure of an AOG building block which prunes syntactically symmetric nodes.

3.1. The Structure of an AOGNet

An AOGNet consists of a predefined number of stages each of which comprises one or more than one AOG building blocks. Fig. 3 shows a 3-stage AOGNet.

As Fig. 2 illustrates, an AOG building block maps an input feature map F with the dimensions $C \times H \times W$ (representing the number of channels, height and width respectively) to an output feature map $\mathbb{F}$ with the dimensions $\mathbb{C} \times \mathbb{H} \times \mathbb{W}$. *We split the input feature map into N groups along feature channels, and then treat it as a "sentence of N words"*. Each word represents a primitive feature map with the dimensionality $c \times H \times W$ in the input, satisfying $C = N \times c$. In implementation, following a common convention, we usually reduce the spatial size and increase the number of channels between consecutive stages for bigger receptive field and greater expressive power. Our AOG building blocks integrates two grammars (see Algorithm 1).

The phrase structure grammar [8, 9, 7, 60, 59, 41]. We consider the following three rules in unfolding the configurations of a sentence with N words:

$$S_{i,j} \to t_{i,j}, \tag{1}$$
$$S_{i,j}(m) \to L_{i,i+m} \cdot R_{i+m+1,j}, \quad 0 \le m < k, \tag{2}$$
$$S_{i,j} \to S_{i,j}(0)|S_{i,j}(1)|\cdots|S_{i,j}(j-i). \tag{3}$$

where $S_{i,j}$ represents a symbol for parsing the sub-sentence starting at the i-th word ($i \in [0, N-1]$) and ending at the j-th word ($j \in [0, N-1], j \ge i$) and its length equals $k = j - i + 1$.

- The first rule is a termination rule which grounds the non-terminal symbol $S_{i,j}$ directly to the corresponding sub-sentence $t_{i,j}$, i.e., a k-gram terminal symbol, which is represented by a **Terminal-node**.
- The second rule is a binary decomposition rule which decomposes the non-terminal symbol $S_{i,j}$ into two child symbols representing a left sub-sentence and a right sub-sentence respectively: $L_{i,i+m}$ and $R_{i+m+1,j}$, both of which are either a non-terminal symbol or a terminal symbol depending on m. It is represented by an **AND-node**, and entails **the concatenation scheme** in forward computation.
- The third rule represents alternative ways of decomposing a symbol $S_{i,j}$, which is represented by an **OR-node**, and entails **summation scheme** in forward computation to "integrate out" the decomposition structures.

```
Input: The total length (or primitive size) $N$.
Output: The AND-OR Graph $\mathcal{G} =< V, E >$
Initialization: Create an OR-node $O_{0,N-1}$ for the
  entire sentence, $V = \{O_{0,N-1}\}, E = \emptyset$, BFS queue
  $Q = \{O_{0,N-1}\}$;
while $Q$ is not empty do
    Pop a node $v_{i,j}$ from the $Q$ and let $k = j - i + 1$;
    if $v_{i,j}$ is an OR-node then
        i) Add a terminal-node $t_{i,j}$, and update
          $V = V \cup \{t_{i,j}\}, E = E \cup \{< v_{i,j}, t_{i,j} >\}$;
        ii) Create AND-nodes $A_{i,j}(m)$ for all valid
          splits $0 \le m < k$;
        $E = E \cup \{< v_{i,j}, A_{i,j}(m) >\}$;
        if $A_{i,j}(m) \notin V$ then
            $V = V \cup \{A_{i,j}(m)\}$;
            Push $A_{i,j}(m)$ to the back of $Q$;
        end
    else if $v_{i,j}$ is an AND-node with split index $m$ then
        Create two OR-nodes $O_{i,i+m}$ and $O_{i+m+1,j}$
          for the two sub-sentence respectively;
        $E = E \cup \{< v_{i,j}(m), O_{i,i+m} >, <$
          $v_{i,j}(m), O_{i+m+1,j} >\}$;
        if $O_{i,i+m} \notin V$ then
            $V = V \cup \{O_{i,i+m}\}$;
            Push $O_{i,i+m}$ to the back of $Q$;
        end
        if $O_{i+m+1,j} \notin V$ then
            $V = V \cup \{O_{i+m+1,j}\}$;
            Push $O_{i+m+1,j}$ to the back of $Q$;
        end
    end
end
Add lateral connections between between
  non-terminal nodes of the same type (AND-node or
  OR-node) with the same length.
```

Algorithm 1: Constructing an AOG building block

The dependency grammar [14, 10, 60]. We introduce dependency grammar to model lateral connections between non-terminal nodes of the same type (AND-node or OR-node) with the same length (i.e., $k = j - i + 1$ in the three rules). As illustrated by the arrows in pink in Fig. 2, we add lateral connections in a simple way: (i) For the set of OR-nodes with $k \in [1, N-1]$, we first sort them based on the starting index i; and (ii) For the set of AND-nodes with $k \in [2, N]$, we first sort them based on the lexical orders of the pairs of starting indexes of the two child nodes. Then, we add sequential lateral connections for nodes in the sorted set either from left to right or from right to left. We usually use opposite lateral connection directions for AND-nodes and OR-nodes to have globally consistent lateral flow from bottom to top in an AOG building block.

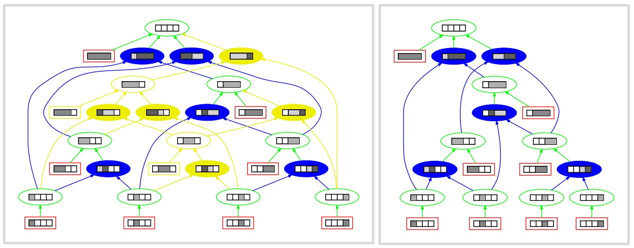
Figure 4. Illustration of simplifying the AOG building blocks by pruning syntactically symmetric child nodes of OR-nodes. *Left:* An AOG building block with full structure consisting of 10 Terminal-nodes, 10 AND-nodes and 10 OR-nodes. Nodes and edges to be pruned are plotted in yellow. *Right:* The simplified AOG building block consisting of 8 Terminal-nodes, 5 AND-nodes and 8 OR-nodes. Here, lateral connections are not shown for clarity. (Best viewed in color)

3.2. Node Operations in an AOGNet

In an AOG building bock, all nodes use the sample type of transofrmation function $\mathcal{T}(\cdot)$ (see Fig. 2). For a node $v_{i,j}$ with length $k = j - i + 1$, denote by $f^v_{i,j}$ its input feature map, and $\mathbf{f}^v_{i,j}$ its output feature map computed by $\mathbf{f}^v_{i,j} = \mathcal{T}(f^v_{i,j})$. We have,

- For a Terminal-node $t_{i,j}$, denote by $F_{i,j}$ the corresponding k-gram chunk in the input feature map F. We have its input $f^t_{i,j} = F_{i,j}$ with the dimensionality $c^v_{i,j} \times H \times W$, and its output $\mathbf{f}^t_{i,j} = \mathcal{T}(F_{i,j})$ with the dimensionality $\mathbf{c}^v_{i,j} \times \mathbb{H} \times \mathbb{W}$, where $c^v_{i,j} = k \times c$ and $\mathbf{c}^v_{i,j} = k \times \frac{\mathbb{C}}{N}$.
- For an AND-node $A_{i,j}(m)$, its input is computed by the concatenation of the outputs of its two syntactic child nodes, $\mathbf{f}^L_{i,i+m}$ and $\mathbf{f}^R_{i+m+1,j}$. We have $f^A_{i,j} = [\mathbf{f}^L_{i,i+m}, \mathbf{f}^R_{i+m+1,j}]$. If it has a lateral child node whose output is denoted by $\mathbf{f}^A_{lateral}$, we add it to $f^A_{i,j}$, i.e., $f^A_{i,j} = [\mathbf{f}^L_{i,i+m}, \mathbf{f}^R_{i+m+1,j}] + \mathbf{f}^A_{lateral}$.
- For an OR-node $O_{i,j}$, its input is the summation of the outputs of its child nodes, $f^O_{i,j} = \sum_{u_{i,j} \in ch(O_{i,j})} \mathbf{f}^u_{i,j}$, where $ch(\cdot)$ be the set of child nodes. If it has a lateral child node whose output is denoted by $\mathbf{f}^O_{lateral}$, we add it to $f^O_{i,j}$, i.e., $f^O_{i,j} = \sum_{u_{i,j} \in ch(O_{i,j})} \mathbf{f}^u_{i,j} + \mathbf{f}^O_{lateral}$.

Where the input and output of an AND-node or an OR-node, $v_{i,j}$, have the same dimensionalities: $\mathbf{c}^v_{i,j} \times \mathbb{H} \times \mathbb{W}$ and $\mathbf{c}^v_{i,j} = k \times \frac{\mathbb{C}}{N}$. In learning and inference, we follow the depth-first search (DFS) order to compute nodes in an AOG building block, which ensures that all the child nodes have been computed when we compute a node v.

3.3. Simplifying AOG Building Blocks

The phrase structure grammar is syntactically redundant since it aims to explore all possible configurations w.r.t. the binary compositional rule. In representation learning, we usually want to increase the feature dimensions of different stages in a network for improving the representational power without increasing the total number of parameters too much. To balance the structural complexity and the feature dimensions in our AOG building blocks, we propose to simplify the structure of AOG building blocks by pruning. The pruning adopts a simple method: We follow the BFS order of nodes, and for each encountered OR-node we only keep the child nodes which do not have left-right syntactically symmetric counterparts in the current set of child nodes. For example, consider the four child nodes of the root OR-node in the left of Fig. 4, the fourth child node is removed since it is symmetric to the second one. Following the BFS, we can extract the pruned AOG building block.

4. Experiments

In this section, we test our AOGNets on three highly competitive image classification benchmarks: CIFAR-10 and CIFAR-100 [24], and ImageNet-1K [38], and on the PASCAL VOC 2007 and 2012 benchmarks [5]. We implemented our AOGNets using PyTorch.

4.1. Implementation Settings

In our experiments, we use simplified AOG building blocks. We also use the same building block for all stages for simplicity. For node operations $\mathcal{T}()$'s, we use the bottleneck variant of Conv-BatchNorm-ReLU, as done in ResNets [15], which adds one 1×1 convolution before and after the operation to first reduce the dimensionality and then expand it back respectively. We notice that we can apply different AOG building blocks and node operations for different types of nodes as long as we can match the dimensions during the computation. We keep them simple in this paper. We leave the exploration of different operators in future work.

The depth of an AOGNet is defined by the largest number of units which have learnable parameters along the paths from the final output to the input data following BFS order. *E.g.*, the longest path in the simplified AOG building block in Fig. 4 is 8, and a bottleneck operation is counted as 3 units, so the depth of the simplified AOG building block is counted as 24. In comparison, to indicate the specifications, AOGNets will be written by AOGNet-*PrimitiveSize*-*(#AOG blocks per stage)*-[*OutputFeatDim*]. *E.g.*, AOGNet-4-(1,1,1,1)-256d represents a 4-stage AOGNet with 1 AOG building block per stage, primitive size being 4, and the final output feature dimension 256.

4.2. Experiments on CIFAR

CIFAR-10 and CIFAR-100 datasets [24], denoted by C10 and C100 respectively, consist of 32×32 color images drawn from 10 and 100 classes. The training and test sets contains $50,000$ and $10,000$ images respectively. We adopt widely used standard data augmentation scheme, random cropping and mirroring, in preparing the training data.

Method	Depth	#Params	FLOPs	C10	C100
ResNet [15]	110	1.7M	0.251G	6.61	-
ResNet (reported by [22])	110	1.7M	0.251G	6.41	27.22
ResNet (pre-activation) [16]	164	1.7M	0.251G	5.46	24.33
	1001	10.2M	-	4.62	22.71
Wide ResNet [55]	16	11.0M	-	4.81	22.07
DenseNet-BC [21] ($k = 12$)	100	0.8M	0.292G	4.51	22.27
AOGNet-4-(1,1,1)-252d	74	**0.78M**	**0.123G**	**4.37**	**20.95**
DenseNet-BC [21] ($k = 24$)	250	15.3M	5.46G	3.62	17.60
AOGNet-4-(1,1,1)-1152d	98	15.8M	**2.4G**	**3.42**	**16.93**
Wide ResNet [55]	28	36.5M	5.24G	4.17	20.50
FractalNet [29]	21	38.6M	-	5.22	23.30
with Dropout/DropPath	21	38.6M	-	4.60	23.73
ResNeXt-29, 8×64d [52]	29	34.4M	3.01G	3.65	17.77
ResNeXt-29, 16×64d [52]	29	68.1M	5.59G	3.58	17.31
DenseNet-BC [21] ($k = 40$)	190	25.6M	9.35G	3.46	17.18
AOGNet-4-(1,1,1)-1444d	98	**24.8M**	3.7G	**3.27**	**16.63**

Table 1. Error rates (%) on the two CIFAR datasets [24]. #Params uses the unit of Million. k in DenseNet refers to the growth rate.

We train AOGNets with stochastic gradient descent (SGD) for 300 epochs with random parameter initialization. The front-end (see Fig. 3) uses a single convolution layer. The initial learning rate is set to 0.1, and is divided by 10 at 150 and 225 epoch respectively. For CIFAR-10, we chose batch size 64 with weight decay 1×10^{-4}, while batch size 128 with weight decay 5×10^{-4} is adopted for CIFAR-100. The momentum is set to 0.9.

Results and Analyses. We summarize the results in Table 1. With smaller model sizes and much reduced computing complexity (FLOPs), our AOGNets obtain better performance than ResNets [15] and some of the variants, ResNeXts [52] and DenseNets [21] consistently on both datasets. Our small AOGNet ($0.78M$) already outperforms the ResNet [15] ($10.2M$) and the WideResNet [55] ($11.0M$). Since the same node operation is used, the improvement must come from the AOG building block structure. Compared with the DenseNets, our AOGNets improve more on C100, and use less than half FLOPs for comparable model sizes. The reason for the reduced FLOPs is that DenseNets apply down-sampling after each Dense block, while our AOGNets sub-sample at Terminal-nodes.

4.3. Experiments on ImageNet-1K

The ILSVRC 2012 classification dataset [38] consists of about 1.2 million images for training, and $50,000$ for validation, from $1,000$ classes. We adopt the same data augmentation scheme (random crop and horizontal flip) for training images as done in [15, 16, 21], and apply a single-crop with size 224×224 at test time. Following the common protocol, we evaluate the top-1 and top-5 classification error rates on the validation set.

We train AOGNets with SGD for 120 epochs and random parameter initialization. The front-end (see Fig. 3) uses three 3×3 Convolution+BatchNorm layers (with stride 2 for the first layer), followed by a 3×3 max pooling layer with stride 2. We used 8 GPUs (NVIDIA V100) in training. The batch size is 128 per GPU (1024 in total). The initial learning rate is 0.4, and the cosine learning rate scheduler [33] is used with weight decay 1×10^{-4} and momentum 0.9.

Method	#Params	FLOPS	top-1	top-5
ResNet-101 [15]	44.5M	8G	23.6	7.1
ResNet-152 [15]	60.2M	11G	23.0	6.7
ResNeXt-50 [52]	25.03M	4.2G	22.2	5.6
ResNeXt-101 ($32 \times 4d$) [52]	44M	8.0G	21.2	5.6
ResNeXt-101 ($64 \times 4d$) [52]	83.9M	16.0G	20.4	5.3
DensetNet-161 [21]	27.9M	7.7G	22.2	-
DensetNet-169 [21]	13.5M	4G	23.8	6.85
DensetNet-264 [21]	33.4M	-	22.2	6.1
ResNeXt-50+SE [19]	25M	4.3G	21.1	5.49
ResNeXt-101+SE [19]	44M	8.0G	20.70	5.01
DPN-68 [3]	12.8M	2.5G	23.57	6.93
DPN-92 [3]	38.0M	6.5G	20.73	5.37
DPN-98 [3]	61.6M	11.7G	20.15	5.15
AOGNet-4-(1,1,2,1)-800d	11.7M	2.19G	22.34	6.16
AOGNet-4-(1,1,3,1)-1400d	40.3M	7.56G	20.08	5.07
AOGNet-4-(1,1,4,1)-1800d	60.2M	12.69G	19.73	4.91

Table 2. The top-1 and top-5 error rates (%) on the ImageNet-1K validation set using single model and single-crop testing.

Method	#Params	FLOPS	top-1	top-5
MobileNetV1 [18]	4.2M	575M	29.4	10.5
SqueezeNext [11]	4.4M	-	30.92	10.6
ShuffleNet (1.5) [58]	3.4M	292M	28.5	-
ShuffleNet (x2) [58]	5.4M	524M	26.3	-
CondenseNet (G=C=4) [20]	4.8M	529M	26.2	8.3
MobileNetV2 [40]	3.4M	300M	28.0	9.0
MobileNetV2 (1.4) [40]	6.9M	585M	25.3	7.5
NASNet-C (N=3) [61]	4.9M	558M	27.5	9.0
AOGNet-4-(1,1,2,1)-608d	4.6M	546M	26.5	8.4

Table 3. The top-1 and top-5 error rates (%) on the ImageNet-1K validation set using single model and single-crop testing.

Results and Analyses. Table 2 shows the results. Our AOGNets are the best among the models with comparable model sizes in comparison in terms of both accuracy. Our small AOGNet ($11.97M$) even outperforms ResNets [15] ($44.5M$ and $60.2M$) by 1.2% and 0.6% respectively. Similarly, we note that our AOGNets use the same bottleneck operation function as ResNets, so the improvement must be contributed by the AOG building block structure. Our AOGNet ($40.3M$) obtains better performance than ResNeXt [52]+SE [19] ($44M$) which represents the most powerful and widely used combination in practice. It also obtains better performance than the best model, DPN [3] ($61.6M$), which indicates that the hierarchical and compositional integration of the DenseNet- and ResNet-aggregation in our AOG building blocks are more effective than the cascade-based integration in the DPN [3]. Our AOGNet ($60.2M$) achieves the best result. We note that the FLOPs of our AOGNets are slightly higher than DPNs since DPNs use ResNeXt operation (i.e., group convolutions). In our on-going experiments, we are testing AOGNets with ResNeXt operation for nodes.

To further evaluate AOGNets for mobile platforms, we

	Method	#params	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
07trainval/07test	ResNet-101*	47.5M	72.3	76.2	77.9	74.6	59.9	53.0	80.8	81.7	85.3	49.0	80.2	64.1	83.7	83.3	76.5	77.8	45.2	73.0	72.0	81.7	71.3
	AOGNet-4-(1,1,2,1)-800d	13.6M	72.1	76.0	77.4	72.6	59.2	55.6	80.4	83.7	85.2	50.1	77.1	63.7	83.8	82.8	78.7	77.8	44.7	71.4	72.5	79.1	70.6
	AOGNet-4-(1,1,3,1)-1400d	43.2M	**75.8**	77.2	84.4	78.0	64.5	60.6	83.5	87.2	85.8	55.4	85.1	65.7	86.0	84.5	80.3	78.9	52.4	75.5	72.6	82.6	75.9
07+12trainval/07test	ResNet-101 [15]	47.5M	76.4	79.8	80.7	76.2	68.3	55.9	85.1	85.3	89.8	56.7	87.8	69.4	88.3	88.9	80.9	78.4	41.7	78.6	79.8	85.3	72.0
	ResNet-101*	47.5M	78.1	81.0	85.4	78.5	71.0	62.7	85.8	86.6	87.7	62.2	84.3	68.9	87.2	86.8	81.3	79.3	51.4	83.4	75.8	85.3	76.6
	AOGNet-4-(1,1,2,1)-800d	13.6M	78.3	80.0	83.1	78.4	71.9	66.0	84.6	87.2	87.5	57.6	84.0	71.6	86.4	87.0	84.5	81.1	51.3	80.1	78.2	87.2	77.6
	AOGNet-4-(1,1,3,1)-1400d	43.2M	**81.0**	87.0	86.3	81.0	72.2	70.9	87.5	88.3	89.3	64.5	85.4	74.0	89.1	88.1	84.6	83.7	55.5	85.9	81.4	87.2	78.8
07+12trainval/12test	ResNet-101*	47.5M	75.1	86.1	82.5	77.6	63.4	54.5	80.6	79.9	91.0	55.8	79.9	56.0	89.5	82.6	83.3	83.1	53.9	79.8	67.4	86.3	69.5
	AOGNet-4-(1,1,2,1)-800d	13.6M	75.2	87.0	82.1	78.4	63.1	56.7	80.0	80.7	91.5	55.1	79.7	59.9	88.8	83.8	82.8	83.4	54.3	79.0	65.8	84.7	67.4
	AOGNet-4-(1,1,3,1)-1400d	43.2M	**78.0**	88.9	82.8	81.3	66.9	62.4	82.6	83.0	92.5	59.6	82.5	61.9	90.9	86.1	84.5	84.5	56.1	83.3	69.7	87.6	71.5

Table 4. Performance comparisons using Average Precision (AP) at the intersection over union (IoU) threshold 0.5 (AP@0.5) in the PASCAL VOC2007 / 2012 dataset. * reported based on our re-implementation using the exactly same PyTorch implementation of Faster R-CNN and PyTorch pretrained ResNet-101 backbone on ImageNet for fair comparisons. The reproduced results of ResNets are better than those reported in the original paper [15].

trained an AOGNet ($4.6M$) and Table 3 shows the comparison results. We obtain performance on par to the popular networks specifically designed for mobile platforms such as the MobileNets [18, 40] and ShuffleNets [58]. Our AOGNet also outperforms the auto-searched network, NASNet [61] (which used around 800 GPUs in search). We note that we used the same AOGNet structures, and thus show the platform-agnostic capability of our AOGNets. This is potentially important and useful for deploying DNNs to different platforms in practice since no extra efforts of hand-crafting or searching neural architectures are entailed. This will be also potentially useful for distilling a small model from a large model if they share the exactly same structure.

4.4. Object Detection on PASCAL VOC

We test our AOGNets in object detection on the PASCAL VOC 2007 and 2012 datasets [5]. We adopt the vanilla Faster R-CNN system [37] and reuse the code in PyTorch [1]. We only substitute the ConvNet backbone with our AOGNets in experiments and keep everything else unchanged for fair comparison. We finetue the AOGNets pretrained on ImageNet. We adopt the provided end-to-end training procedure to train the region proposal network (RPN) and R-CNN jiontly. The first three stages are shared by RPN and R-CNN, and the last stage is used as the head classifier for region-of-interest (RoI) prediction. We fix all parameters pretrained on ImageNet before stage 1 (inclusive) in training. We follow the standard evaluation metrics Average Precision (AP) and mean of AP (mAP) in evaluation [5]. Table 4 shows the detection results and comparisons. Our AOGNets obtain better mAP than ResNet-101 by around 3% consistently. When trained using the $07 + 12$ trainval dataset, our small AOGNet-backboned detector ($13.6M$) already slightly outperforms the ResNet-backboned one ($47.5M$), which further shows the effectiveness of the AOG building blocks in object detection tasks.

4.5. Ablation Study

We conduct an ablation study which investigates the effects of *(i) RS:* Removing Symmetric child nodes of OR-

[1]https://github.com/jwyang/faster-rcnn.pytorch

Method	#Params	FLOPS	CIFAR10	CIFAR100
AOGNet	4.24M	0.65G	3.75	19.20
AOGNet+LC	4.24M	0.65G	3.70	19.09
AOGNet+RS	4.23M	0.70G	3.57	18.64
AOGNet+RS+LC	4.23M	0.70G	**3.52**	**17.99**

Table 5. An ablation study of our AOGNets using the mean error rate across 5 runs. In the first two rows, the AOGNets use full structure, and the pruned structure in the last two rows. The feature dimensions of node operations are accordingly specified to keep model sizes comparable.

nodes in the pruned AOG building blocks, and of *(ii) LC:* adding Lateral Connections for dependency grammars. As Table 5 shows, the two components, RS and LC, improve performance. The results are consistent with our design intuition and principles. The RS component facilitates higher feature dimensions due to the reduced structural complexity, and the LC component increases the effective depth of nodes on the lateral flows.

5. Conclusions

This paper presented a method of learning deep compositional grammatical architectures which are capable of harnessing the best of grammars and deep neural networks for visual recognition. AND-OR Grammars (AOG) comprising phrase structure grammars and dependency grammars are utilized to design building blocks. The resulting models are called AOGNets. An AOGNet consists of a number of stages of AOG building blocks. AOGNets are tested on three highly competitive and widely used image classification benchmarks: the CIFAR-10 dataset and the CIFAR-100 dataset [24], and the ImageNet-1K dataset [38]. Our AOGNets obtain better performance consistently than ResNets [15] and most variants, ResNeXts [52], DenseNets [21] and DualPathNets [3] when model sizes are comparable. AOGNets are also tested in object detection on the PASCAL VOC 2007 and 2012 [5] using the vanilla Faster R-CNN [37] system, and obtain better performance by about 3% mAP than ResNets [15].

Acknowledgment. This work is supported by ARO award W911NF1810295, ARO DURIP award W911NF1810209 and NSF IIS 1822477. Some early experiments in this work used the Extreme Science and Engineering Discovery Environment (XSEDE) [48] at the SDSC Comet GPU Nodes (Comet GPU) through allocation IRI180024 (XSEDE is supported by National Science Foundation grant number ACI-1548562).

References

[1] S. Arora, A. Bhaskara, R. Ge, and T. Ma. Provable bounds for learning some deep representations. In *Proceedings of the 31th International Conference on Machine Learning, ICML*, pages 584–592, 2014. 1

[2] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference, BMVC*, 2014. 4

[3] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. *arXiv preprint arXiv:1707.01629*, 2017. 1, 2, 4, 7, 8

[4] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *CoRR*, abs/1808.05377, 2018. 4

[5] M. Everingham, S. M. Eslami, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vision (IJCV)*, 111(1):98–136, Jan. 2015. 1, 4, 6, 8, 9

[6] P. F. Felzenszwalb. Object detection grammars. In *IEEE International Conference on Computer Vision Workshops, ICCV*, page 691, 2011. 3, 4

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, 32(9):1627–1645, Sept. 2010. 3, 5

[8] K. S. Fu and J. E. Albus, editors. *Syntactic pattern recognition : applications*. Communication and cybernetics. Springer-Verlag, Berlin, New York, 1977. 3, 5

[9] S. Geman, D. Potter, and Z. Y. Chi. Composition systems. *Quarterly of Applied Mathematics*, 60(4):707–736, 2002. 2, 3, 5

[10] D. George, W. Lehrach, K. Kansky, M. Lázaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang, A. Lavin, and D. S. Phoenix. A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science*, 2017. 2, 3, 4, 5

[11] A. Gholami, K. Kwon, B. Wu, Z. Tai, X. Yue, P. Jin, S. Zhao, and K. Keutzer. Squeezenext: Hardware-aware neural network design. *arXiv preprint arXiv:1803.10615*, 2018. 7

[12] U. Grenander and M. Miller. Pattern theory: From representation to inference. 2

[13] D. Han, J. Kim, and J. Kim. Deep pyramidal residual networks. *IEEE CVPR*, 2017. 2, 3

[14] D. G. Hays. Dependency theory: A formalism and some observations. *Language*, 40(4):511–525, 1964. 3, 5

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 3, 4, 6, 7, 8, 9

[16] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, pages 630–645, 2016. 7

[17] G. Hinton. What is wrong with convolutional neural nets? *the 2017 - 2018 Machine Learning Advances and Applications Seminar presented by the Vector Institute at U of Toronto, https://www.youtube.com/watch?v=Mqt8fs6ZbHk*, August 17, 2017. 1

[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 7, 8

[19] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017. 4, 7

[20] G. Huang, S. Liu, L. van der Maaten, and K. Q. Weinberger. Condensenet: An efficient densenet using learned group convolutions. *group*, 3(12):11, 2017. 7

[21] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 3, 4, 7, 8

[22] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, pages 646–661, 2016. 4, 7

[23] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In D. Blei and F. Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 448–456. JMLR Workshop and Conference Proceedings, 2015. 1, 4

[24] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. 4, 6, 7, 8

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems (NIPS)*, pages 1106–1114, 2012. 1, 4

[26] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 2

[27] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015. 4

[28] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *CoRR*, abs/1604.00289, 2016. 4

[29] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *CoRR*, abs/1605.07648, 2016. 4, 7

[30] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1, 4

[31] L. Lin, T. Wu, J. Porway, and Z. Xu. A stochastic graph grammar for compositional object representation and recognition. *Pattern Recognition*, 42(7):1297–1307, 2009. 4

[32] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, abs/1312.4400, 2013. 4

[33] I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016. 7

[34] D. Mumford. Grammar isn't merely part of language. http://www.dam.brown.edu/people/mumford/blog/2016/grammar.html. 3

[35] D. Mumford and A. Desolneux. Pattern theory, the stochastic analysis of real world signals. 2

[36] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. *CoRR*, abs/1603.06937, 2016. 4

[37] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*, 2015. 1, 4, 8, 9

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vision (IJCV)*, 115(3):211–252, 2015. 4, 6, 7, 8

[39] S. Sabour, N. Frosst, and G. E Hinton. Dynamic Routing Between Capsules. *ArXiv e-prints*, Oct. 2017. 1

[40] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 7, 8

[41] X. Song, T. Wu, Y. Jia, and S. Zhu. Discriminatively trained and-or tree models for object detection. In *Proceedings of 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3278–3285, 2013. 1, 3, 4, 5

[42] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015. 4

[43] P. D. Summers. A methodology for LISP program construction from examples. *J. ACM*, 24(1):161–175, 1977. 4

[44] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. 1, 2, 3, 4

[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9, 2015. 4

[46] W. Tang, P. Yu, and Y. Wu. Deeply learned compositional models for human pose estimation. In *ECCV (3)*, volume 11207 of *Lecture Notes in Computer Science*, pages 197–214. Springer, 2018. 4

[47] W. Tang, P. Yu, J. Zhou, and Y. Wu. Towards a unified compositional model for visual pattern modeling. In *ICCV*, pages 2803–2812. IEEE Computer Society, 2017. 4

[48] J. Towns, T. Cockerill, M. Dahan, I. T. Foster, K. P. Gaither, A. S. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr. XSEDE: accelerating scientific discovery. *Computing in Science and Engineering*, 16(5):62–74, 2014. 9

[49] J. Wang, Z. Wei, T. Zhang, and W. Zeng. Deeply-fused nets. *CoRR*, abs/1605.07716, 2016. 4

[50] X. Wang and L. Lin. Dynamical and-or graph learning for object shape modeling and detection. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 242–250, 2012. 4

[51] T. Wu, Y. Lu, and S. Zhu. Online object tracking, learning and parsing with and-or graphs. *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, DOI: 10.1109/TPAMI.2016.2644963, 2016. 3

[52] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016. 1, 2, 3, 4, 7, 8

[53] Q. Yao, M. Wang, H. J. Escalante, I. Guyon, Y. Hu, Y. Li, W. Tu, Q. Yang, and Y. Yu. Taking human out of learning applications: A survey on automated machine learning. *CoRR*, abs/1810.13306, 2018. 4

[54] F. Yu, D. Wang, and T. Darrell. Deep layer aggregation. In *CVPR*, 2018. 2, 4

[55] S. Zagoruyko and N. Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*, 2016. 4, 7

[56] T. Zhang, G. Qi, B. Xiao, and J. Wang. Interleaved group convolutions. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 4383–4392, 2017. 2, 4

[57] X. Zhang, Z. Li, C. C. Loy, and D. Lin. Polynet: A pursuit of structural diversity in very deep networks. *CoRR*, abs/1611.05725, 2016. 4

[58] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017. 7, 8

[59] L. Zhu, Y. Chen, Y. Lu, C. Lin, and A. L. Yuille. Max margin AND/OR graph learning for parsing the human body. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 1, 3, 4, 5

[60] S. C. Zhu and D. Mumford. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, 2(4):259–362, 2006. 1, 3, 4, 5

[61] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2(6), 2017. 7, 8