# AOGNets: Compositional Grammatical Architectures for Deep Learning

Xilai Li[†], Xi Song[§] and Tianfu Wu[†,‡*]

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. We utilize AND-OR Grammar (AOG) [52, 71, 70] as network generator 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 reuse. It provides a unified framework for the best practices developed in state-of-the-art DNNs. In experiments, AOGNet is tested in the ImageNet-1K classification benchmark and the MS-COCO object detection and segmentation benchmark. In ImageNet-1K, AOGNet obtains better performance than ResNet [21] and most of its variants, ResNeXt [63] and its attention based variants such as SENet [24], DenseNet [26] and DualPathNet [6]. AOGNet also obtains the best model interpretability score using network dissection [3]. AOGNet further shows better potential in adversarial defense. In MS-COCO, AOGNet obtains better performance than the ResNet and ResNeXt backbones in Mask R-CNN [20].*

## 1. Introduction

### 1.1. Motivation and Objective

Recently, deep neural networks (DNNs) [35, 30] have improved prediction accuracy significantly in many vision tasks, and have obtained superhuman performance in image classification tasks [21, 55, 26, 6]. Much of these

---

*T. Wu is the corresponding author. [§]X. Song is an independent researcher. The code and models are available at https://github.com/iVMCL/AOGNets
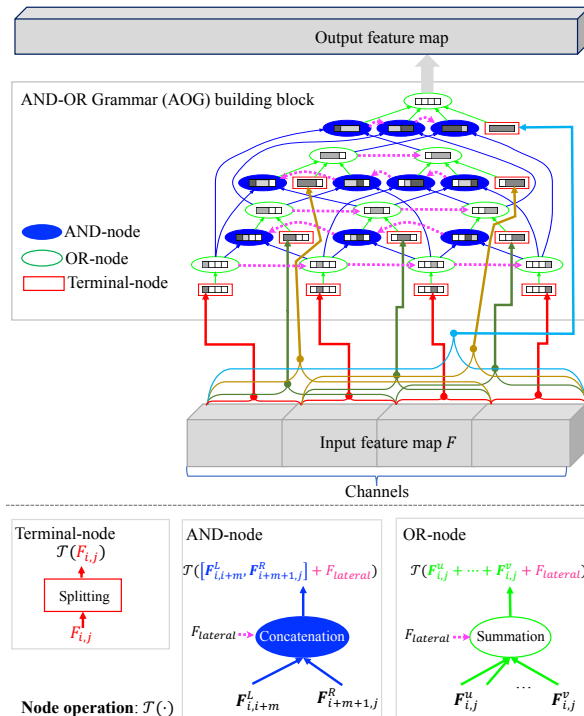


Figure 1. Illustration of our AOG building block for grammar-guided network generator. The resulting networks, AOGNets obtain 80.18% top-1 accuracy with 40.3M parameters in ImageNet, significantly outperforming ResNet-152 (77.0%, 60.2M), ResNeXt-101 (79.6%, 83.9M), DenseNet-Cosine-264 (79.6%, ~73M) and DualPathNet-98 (79.85%, 61.6M). See text for details. (Best viewed in color)

progress are achieved mainly through engineering network architectures which jointly address two issues: increasing representational power by going either deeper or wider, and maintaining the feasibility of optimization using back-propagation with stochastic gradient descent (i.e., the vanishing and/or exploding gradient problems). The dramatic success does not necessarily speak to its sufficiency. Hinton recently pointed out: according to recent neuroscientific research, these artificial networks do not contain enough levels of structure [22, 50]. In this paper, we are interested in **grammar-guided network generators** (Fig. 1).

Neural architecture design and search can be posed as a combinatorial search problem in a product space comprising two sub-spaces (Fig. 2 (a)): (i) 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. (ii) The node operation space which consists of all possible transformation functions for implementing nodes in a DAG, such as Conv+BN [28]+ReLU [30] with different kernel sizes and feature channels.

The structure space is almost unbounded, and the node operation space for a given structure is also combinatorial. Neural architecture design and search is a challenging 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. 2 (b), to mitigate the difficulty, neural architecture design and search have been simplified to design or search a building block structure. Then, a DNN consists of a predefined number of stages each of which has a small number of building blocks. This stage-wise building-block based design is also supported by the theoretical study in [1] under some assumptions. Fig. 2 (c) shows examples of some popular building blocks with different structures. Two questions arise naturally:

- Can we unify the best practices used by the popular building blocks in a simple and elegant framework? More importantly, can we generate building blocks and thus networks in a principled way to effectively unfold the space (Fig. 2 (a)) ? (If doable)
- Will the unified building block/network generator improve performance on accuracy, model interpretability and adversarial robustness without increasing model complexities and computational costs? If yes, the potential impacts shall be broad and deep for representation learning in numerous practical applications.

To address the above questions, we first need to understand the underlying wisdom in designing better network architectures: It usually lies in finding network structures which can support flexible and diverse information flows for exploring new features, reusing existing features in previous layers and back-propagating learning signals (e.g., gradients). Then, what are the key principles that we need to exploit and formulate such that we can effectively and efficiently unfold the structure space in Fig. 2 (a) in a way better than existing networks? *Compositionality, reconfigurability and lateral connectivity* are well-known principles in cognitive science, neuroscience and pattern theory [12, 44, 17, 13, 31, 13]. They are fundamental for the remarkable capabilities possessed by humans, of learning rich knowledge and adapting to different environments, especially in vision and language. They have not been, however, fully and explicitly integrated in DNNs.

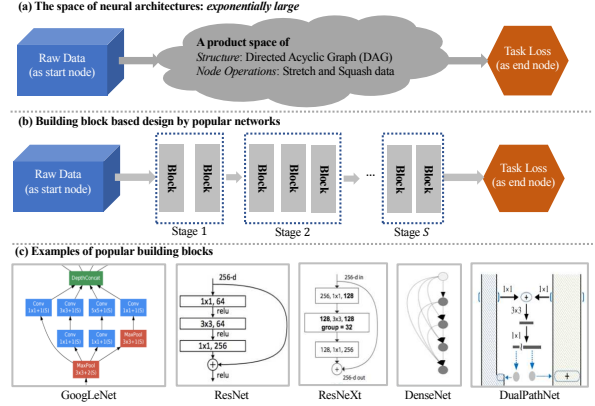In this paper, we presents **compositional grammati-**



Figure 2. Illustration of (a) the space of neural architectures, (b) the building block based design, and (c) examples of popular building blocks in GoogLeNet [55], ResNet [21], ResNeXt [63], DenseNet [26] and DualPathNets [6]. See text for details.

**cal architectures** that realize compositionality, reconfigurability and lateral connectivity for building block design in a principled way. We utilize AND-OR Grammars (AOG) [52, 71, 70] and propose AOG building blocks that unify the best practices developed in existing popular building blocks. Our method deeply integrates hierarchical and compositional grammars and DNNs for harnessing the best of both worlds in deep representation learning.

Why grammars? Grammar models are well known in both natural language processing and computer vision. Image grammar [71, 9, 70, 12] 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. As David Mumford pointed out, "*Grammar 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.*" [43]. Our proposed AOG building block is highly expressive for analyzing sensory input and bridges the performance gap between grammars and DNNs. It also enables flexible and diverse network structures to address Hinton's quest on improving structural sufficiency in DNNs [22].

## 1.2. Method Overview

We first summarize the best practices in existing building blocks, and then briefly overview our proposed AOG building block (Fig. 1) and how it unifies the existing ones.

Existing building blocks usually do not fully implement the three principles (compositionality, reconfigurability and lateral connections).

- InceptionNets or GoogLeNets [55] embodies a split-transform-aggregate heuristic in a shallow feed-forward
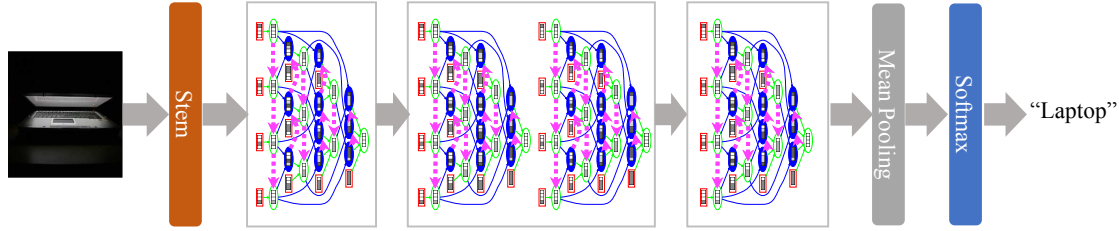
Figure 3. Illustration of a 3-stage AOGNet with 1 AOG building bock in the 1st and 3rd stage, and 2 AOG building blocks in the 2nd stage. Note that different stages can use different AOG building blocks. We show the same one for simplicity. The stem can be either a vanilla convolution or convolution+MaxPooling. (Best viewed in color)

way for feature exploration, which is inspired by the network-in-network design [39] and the theoretical study on stage-wise design [1]. However, the filter numbers and sizes are tailored for each individual transformation, and the modules are customized stage-by-stage. Interleaved group convolutions [67] share the similar spirit, but use simpler scheme.

- ResNets [21] provide a simple yet effective solution, inspired by the Highway network [53], that enables networks to enjoy going either deeper or wider without sacrificing the feasibility of optimization. From the perspective of representation learning, skip-connections within a ResNet [21] contributes to effective feature reuse. They do not, however, realize the split component as done in GoogLeNets.
- ResNeXts [63] add the spit component in ResNets and address the drawbacks of the Inception modules using group convolutions in the transformation.
- Deep Pyramid ResNets [18] gradually increase feature channels between building blocks, instead of increasing feature channels sharply at each residual unit with down-sampling in vanilla ResNets.
- DenseNets [26] explicitly differentiate between information that is added to the network and information that is preserved. Dense connections with feature maps being concatenated together are used, which are effective for feature exploration, but lack the capability of feature reuse as done in ResNets.
- Dual Path Networks (DPN) [6] utilize ResNet blocks and DenseNet blocks in parallel to balance feature reuse and feature exploration.
- Deep Layer Aggregation networks (DLA) [65] iteratively and hierarchically aggregate the feature hierarchy when stacking the building blocks such as the ResNet ones.

Our AOG building block is hierarchical, compositional and reconfigurable with lateral connections by design. As Fig. 1 shows, 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) [11, 12, 10, 71, 70, 52] and a dependency grammar (horizontal connections in pink in Fig. 1) [19, 71, 13] in bottom-up parsing the "sentence"

for better feature exploration and reuse: (i) Phrase structure grammar is a 1-D special case of the method presented in [52, 62]. 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. (ii) Dependency grammar 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-BN-ReLU) to its input, and 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* implements composition, whose input is computed by concatenating features of its syntactic child nodes, and adding the lateral connection if present.
- An *OR-node* represents alternative compositions, whose input is the element-wise sum of features of its syntactic child nodes and the lateral connection if present.

Our AOG building block unifies the best practices developed in popular building blocks in that,

- *Terminal-nodes* implement the split-transform heuristic (or group convolutions) as done in GoogLeNets [55] and ResNeXts [63], 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) [26] for feature exploration.
- *OR-nodes* implement ResNet-like aggregation (i.e., summation) [21] for feature reuse.
- *The hierarchy* facilitates gradual increase of feature channels as in Deep Pyramid ResNets [18], and also leads to good balance between depth and width of networks.
- *The compositional structure* provides much more flexible information flows than DPN [6] and the DLA [65].
- *The lateral connections* induce feature diversity and increase the effective depth of nodes along the path without introducing extra parameters.

We stack AOG building blocks to form a deep AOG network, called **AOGNet**. Fig. 3 illustrates a 3-stage AOGNet. Our AOGNet utilizes two nice properties of grammars: (i) The flexibility and simplicity of constructing different network structures 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.

## 2. Related Work and Our Contributions

Network architectures are the foundation for improving performance of DNNs. We focus on hand-crafted architectures in this section. Related work on neural architecture search is referred to the survey papers [8, 64].

**Hand-crafted network architectures.** After more than 20 years since the seminal work 5-layer LeNet5 [35] was proposed, the recent resurgence in popularity of neural networks was triggered by the 8-layer AlexNet [30] with breakthrough performance on ImageNet [49] 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 [4] proposed a 19-layer network with insights on using multiple successive layers of small filters (e.g., $3 \times 3$) A special case, $1 \times 1$ convolution, was proposed in the network-in-network [39] for reducing or expanding feature dimensionality between consecutive layers, and have been widely used in many networks. The 22-layer GoogLeNet [56] introduced the first inception module and a bottleneck scheme implemented with $1 \times 1$ 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 [53] and popularized by the ResNets [21], especially when combined with the batch normalization (BN) [28]. More than 100 layers are popular design in the recent literature [21, 55], as well as even more than 1000 layers trained on large scale datasets such as ImageNet [27, 68]. The Fractal Net [34] and deeply fused networks [60] 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 [66, 63, 67]. Going beyond the first-order skip-connections in ResNets, DenseNets [26] proposed a densely connected network architecture with concatenation scheme for feature reuse and exploration, and DPNs [6] proposed to combine residuals and densely connections in an alternating way for more effective feature exploration and reuse. DLA networks [65] 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 [24] is a recently proposed simple yet effective method focusing on channel-wise encoding. The Hourglass network [45] 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 [55], the deeply fused nets [60] and the DLA [65].

**Grammars.** A general framework of image grammar was proposed in [71]. Object detection grammar was the dominant approaches for object detection [9, 70, 52, 38, 36, 37], and has recently been integrated with DNNs [57, 58, 5]. Probabilistic program induction [54, 32, 33] 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 [13] have been proposed with better data efficiency in learning which adopts the AND-OR grammar framework [71], showing great potential of grammars in developing general AI systems.

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

- It proposes compositional grammatical architectures for deep learning and presents deep AND-OR Grammar networks (AOGNets). AOGNets facilitate both feature exploration and feature reuse in a hierarchical and compositional way. AOGNets unify the best practices developed by state-of-the-art DNNs such as GoogLeNets, ResNets, ResNeXts, DenseNets, DPN, and DLA. To our best knowledge, it is the first work of designing grammar-guided network generators.
- It obtains better performance than many state-of-the-art networks in the ImageNet-1K classification benchmark and MS-COCO object detection and segmentation benchmark. It also obtains better model interpretability and shows greater potential for adversarial defense.

## 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 (Fig. 3) consists of a predefined number of stages each of which comprises one or more than one AOG building blocks. As Fig. 1 illustrates, an AOG building block maps an input feature map $F_{in}$ with the dimensions $D_{in} \times H_{in} \times W_{in}$ (representing the number of channels, height and width respectively) to an output feature map $F_{out}$ with the dimensions $D_{out} \times H_{out} \times W_{out}$. *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 slice of the input feature map with $\frac{D_{in}}{N} \times H_{in} \times W_{in}$. Our AOG building block is constructed by a simple algorithm (Algorthm 1) which integrates two grammars.

**The phrase structure grammar** [11, 12, 10, 71, 70, 52]. Let $S_{i,j}$ be a non-terminal symbol representing the sub-sentence starting at the $i$-th word ($i \in [0, N-1]$) and ending at the $j$-th word ($j \in [0, N-1], j \geq i$) with the length $k = j - i + 1$. We consider the following three rules in parsing a sentence:

$$S_{i,j} \rightarrow t_{i,j}, \tag{1}$$

$$S_{i,j}(m) \rightarrow [L_{i,i+m} \cdot R_{i+m+1,j}], \quad 0 \leq m < k, \tag{2}$$

$$S_{i,j} \rightarrow S_{i,j}(0)|S_{i,j}(1)|\cdots|S_{i,j}(j-i). \tag{3}$$

where we have,

- 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, denoted by $[L \cdot R]$, which decomposes a non-terminal symbol $S_{i,j}$ into two child non-terminal symbols representing a left sub-sentence and a right sub-sentence, $L_{i,i+m}$ and $R_{i+m+1,j}$ respectively. It is represented by an **AND-node**, and entails **the concatenation scheme** in forward computation to match feature channels.
- The third rule represents alternative ways of decomposing a non-terminal symbol $S_{i,j}$, denoted by $A|B|C$, which is represented by an **OR-node**, and can utilize **summation scheme** in forward computation to "integrate out" the decomposition structures.

**The dependency grammar** [19, 13, 71]. 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 $k$. As illustrated by the arrows in pink in Fig. 1, *we add lateral connections in a straightforward 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 vice versa. We use opposite lateral connection directions for AND-nodes and OR-nodes iteratively to have globally consistent lateral flow from bottom to top in an AOG building block.

### 3.2. Node Operations in an AOGNet

In an AOG building bock, all nodes use the same type of transformation function $\mathcal{T}(\cdot)$ (see Fig. 1). For a node $v$, denote by $f_{in}(v)$ its input feature map, and then its output feature map is computed by $f_{out}(v) = \mathcal{T}(f_{in}(v))$. For a Terminal-node $t$, it is straightforward to apply the transformation using $f_{in}(t) = F_{in}(t)$ where $F_{in}(t)$ is the $k$-gram

---

**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
    $0 \leq m < k$;
    $E = E \cup \{< v_{i,j}, A_{i,j}(m) >\}$;
    **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 (see text for detail).

**Algorithm 1:** Constructing an AOG building block

slice from the input feature map of the AOG building block. For AND-nodes and OR-nodes, we have,

- For an AND-node $A$ with two child nodes $L$ and $R$, its input $f_{in}(A)$ is first computed by the concatenation of the outputs of the two child nodes, $f_{in}(A) = [f_{out}(L) \cdot \lambda_L, f_{out}(R) \cdot \lambda_R]$. If it has a lateral node whose output is denoted by $f_{out}(A_{lateral})$, we add it and get $f_{in}(A) = [f_{out}(L) \cdot \lambda_L, f_{out}(R) \cdot \lambda_R] + f_{out}(A_{lateral}) \cdot \lambda_{lateral}$.
- For an OR-node $O$, its input is the summation of the outputs of its child nodes (including the lateral node if present), $f_{in}(O) = \sum_{u \in ch(O)} f_{out}(u) \cdot \lambda_u$, where $ch(\cdot)$ represents the set of child nodes.

Where $\lambda_L, \lambda_R, \lambda_{lateral}$ and $\lambda_u$'s are weights (see details in Section 4.1). Node inputs are computed following the syntactical structure of AOG building block to ensure that feature dimensions and spatial sizes match in the concatenation and summation. 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 $v$.
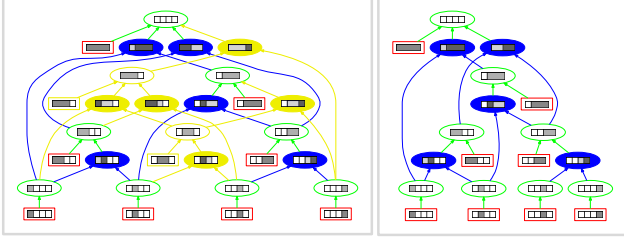
Figure 4. Illustration of simplifying the AOG building blocks by pruning syntactically symmetric child nodes of OR-nodes. *Left:* An AOG building block with full structure consisting of 10 Terminal-nodes, 10 AND-nodes and 10 OR-nodes. Nodes and edges to be pruned are plotted in yellow. *Right:* The simplified AOG building block consisting of 8 Terminal-nodes, 5 AND-nodes and 8 OR-nodes. (Best viewed in color)

### 3.3. Simplifying AOG Building Blocks

The phrase structure grammar is syntactically redundant since it unfolds all possible configurations w.r.t. the binary composition rule. In representation learning, we also want to increase the feature dimensions of different stages in a network for better representational power, but try not to increase the total number of parameters significantly. **To balance the structural complexity and the feature dimensions** of our AOG building block, we propose to simplify the structure of an AOG building block by pruning some syntactically redundant nodes. As illustrated in Fig. 4, the pruning algorithm is simple: Given a full structure AOG building block, we start with an empty simplified block. We first add the root OR-node into the simplified block. Then, we follow the BFS order of nodes in the full structure block. 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 in the simplified block. For encountered AND-nodes and Terminal-nodes, we add them to the simplified block. The pruning algorithm can be integrated into Algorithm 1. 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.

## 4. Experiments

Our AOGNet is tested in the ImageNet-1K classification benchmark [49] and the MS-COCO object detection and segmentation benchmark [40]. [1]

### 4.1. Implementation Settings and Details

We use simplified AOG building blocks. For the node operation $\mathcal{T}()$, we use the bottleneck variant of Conv-BN-ReLU proposed in ResNets [21], which adds one $1 \times 1$ convolution before and after the operation to first reduce feature dimension and then expand it back. More specif-

ically, we have $\mathcal{T}(x) = ReLU(x + T(x))$ for an input feature map $x$ where $T()$ represents a sequence of primitive operations, Conv1x1-BN-ReLU, Conv3x3-BN-ReLU and Conv1x1-BN. If Dropout [30] is used with drop rate $p \in (0, 1)$, we add it after the last BN, i.e., $\mathcal{T}(x) = ReLU(x + Dropout(T(x), p))$

*Handling double-counting due to the compositional DAG structure and lateral connections.* First, in our AOG building block, some nodes will have multiple paths to reach the root OR-node due to the compositional DAG structure. Since we use the skip connection in the node operation $\mathcal{T}()$, the feature maps of those nodes with multiple paths will be double-counted at the root OR-node. Second, if a node $v$ and its lateral node $v_{lateral}$ share a parent node, we also need to handle double-counting in the skip connection. Denote by $n(v)$ the number of paths between $v$ and the root OR-node, which can be counted during the building block construction (Algorithm 1). Consider an AND-node $A$ with two syntactic child node $L$ and $R$ and the lateral node $A_{lateral}$, we compute two different inputs, one for the skip connection, $f_{in}^{skip}(A) = [f_{out}(L) \cdot \frac{n(A)}{n(L)}, f_{out}(R) \cdot \frac{n(A)}{n(R)}]$ if $A$ and $A_{lateral}$ share a parent node and $f_{in}^{skip}(A) = [f_{out}(L) \cdot \frac{n(A)}{n(L)}, f_{out}(R) \cdot \frac{n(A)}{n(R)}] + f_{out}(A_{lateral}) \cdot \frac{n(A)}{n(A_{lateral})}$ otherwise, and the other for $T()$, $f_{in}^{T}(A) = [f_{out}(L), f_{out}(R)] + f_{out}(A_{lateral})$. The transformation for node $A$ is then implemented by $\mathcal{T}(A) = ReLU(f^{skip}(A) + T(f_{in}^{T}(A)))$. Similarly, we can set $\lambda_u$'s in the OR-node operation. We note that we can also treat $\lambda$'s as unknown parameters to be learned end-to-end.

### 4.2. Image Classification in ImageNet-1K

The ILSVRC 2012 classification dataset [49] 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 [21, 26], and apply a single-crop with size $224 \times 224$ at test time. Following the common protocol, we evaluate the top-1 and top-5 classification error rates on the validation set.

**Model specifications.** We test three AOGNets with different model complexities. In comparison, we use the model size as the name tag for AOGNets (e.g., AOGNet-12M means the AOGNet has 12 million parameters or so). The stem (see Fig. 3) uses three Conv3x3-BN layers (with stride 2 for the first layer), followed by a $2 \times 2$ max pooling layer with stride 2. All the three AOGNets use four stages. Within a stage, we use the same AOG building block, while different stages may use different blocks. A stage is then specified by $N_n$ where $N$ is primitive size (Algorithm 1) and $n$ the number of blocks. The filter channels are defined by a 5-tuple for specifying the input and output dimensions for the 4 stages. The detailed specifications

---

[1] Due to space limit, we will present results for CIFAR-10 and CIFAR-100 [29] and other datasets in our code GitHub repository.

| Method | #Params | FLOPS | top-1 | top-5 |
|---|---|---|---|---|
| ResNet-101 [21] | 44.5M | 8G | 23.6 | 7.1 |
| ResNet-152 [21] | 60.2M | 11G | 23.0 | 6.7 |
| ResNeXt-50 [63] | 25.03M | 4.2G | 22.2 | 5.6 |
| ResNeXt-101 ($32\times4d$) [63] | 44M | 8.0G | 21.2 | 5.6 |
| ResNeXt-101 ($64\times4d$) [63] | 83.9M | 16.0G | 20.4 | 5.3 |
| ResNeXt-101 + BAM [46] | 44.6M | 8.05G | 20.67 | - |
| ResNeXt-101 + CBAM [61] | 49.2M | 8.0G | 20.60 | - |
| ResNeXt-50+SE [24] | 27.7M | 4.3G | 21.1 | 5.49 |
| ResNeXt-101+SE [24] | 48.9M | 8.46G | 20.58 | 5.01 |
| DensetNet-161 [26] | 27.9M | 7.7G | 22.2 | - |
| DensetNet-169 [26] | $\sim 13.5M$ | $\sim 4G$ | 23.8 | 6.85 |
| DensetNet-264 [26] | $\sim 33.4M$ | - | 22.2 | 6.1 |
| DensetNet-cosine-264 [47] | $\sim 73M$ | $\sim 26G$ | 20.4 | - |
| DPN-68 [6] | 12.8M | 2.5G | 23.57 | 6.93 |
| DPN-92 [6] | 38.0M | 6.5G | 20.73 | 5.37 |
| DPN-98 [6] | 61.6M | 11.7G | 20.15 | 5.15 |
| AOGNet-12M | 11.9M | 2.36G | 22.28 | 6.14 |
| AOGNet-40M | 40.3M | 8.86G | 19.82 | 4.88 |
| AOGNet-60M | 60.7M | 14.36G | **19.34** | **4.78** |

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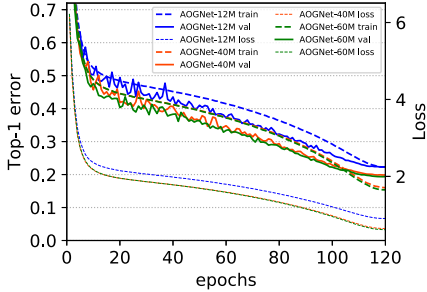


Figure 5. Plots of top-1 error rates and training losses of the three AOGNets in ImageNet. (Best viewed in color and magnification)
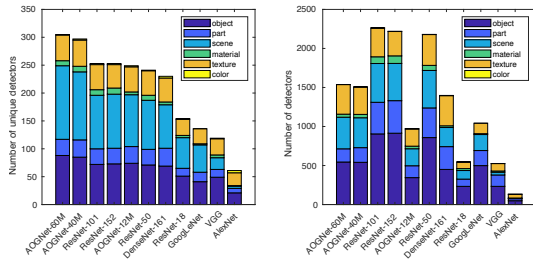


Figure 6. Comparisons of model interpretability using the network dissection method [3] on ImageNet pretrained networks.

of the three AOGNets are: AOGNet-12M uses stages of $(2_2, 4_1, 4_3, 2_1)$ with filter channels $(32, 128, 256, 512, 936)$, AOGNet-40M uses stages of $(2_2, 4_1, 4_4, 2_1)$ with filter channels $(60, 240, 448, 968, 1440)$, and AOGNet-60M uses stages of $(2_2, 4_2, 4_5, 2_1)$ withe filter channels $(64, 256, 512, 1160, 1400)$.

**Training settings.** We adopt random parameter initialization for filter weights. For Batch Normalization (BN) layers, we use $0$ to initialize all offset parameters. We use $1$ to initialize all scale parameters except for the last BN layer in each $\mathcal{T}()$ where we initialize the scale parameter by $0$ as done in [16]. We use Dropout [30] with drop rate

| Method | #Params | $\epsilon = 0.1$ | $\epsilon = 0.3$ | clean |
|---|---|---|---|---|
| ResNet-101 | 44.5M | 12.3 | 0.40 | 77.37 |
| ResNet-152 | 60.2M | 16.3 | 0.85 | 78.31 |
| DenseNet-161 | 28.7M | 13.0 | 2.1 | 77.65 |
| AOGNet-12M | 12.0M | 18.1 | 1.4 | 77.72 |
| AOGNet-40M | 40.3M | 28.3 | 2.2 | 80.18 |
| AOGNet-60M | 60.1M | 30.2 | 2.6 | 80.66 |

Table 2. Top-1 accuracy comparisons under white-box adversarial attack using 1-step FGSM [15] with the Foolbox toolkit [48].

$0.1$ in the last two stages. We use 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 [41] is used with weight decay $1 \times 10^{-4}$ and momentum $0.9$. We train AOGNet with SGD for 120 epochs which include 5 epochs for linear warm-up following [16].

**Results and Analyses: AOGNets obtain the best accuracy and model interpretability.** Table 1 shows the results, and Fig. 5 shows plots for the top-1 error rates and training losses. Our AOGNets are the best among the models with comparable model sizes in comparison in terms of top-1 and top-5 accuracy. Our small AOGNet-12M outperforms ResNets [21] ($44.5M$ and $60.2M$) by $1.32\%$ and $0.72\%$ respectively. *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-40M obtains better performance than all other methods in comparison, including ResNeXt-101 [63]+SE [24] ($48.9M$) which represents the most powerful and widely used combination in practice. AOGNet-40M also obtains better performance than the runner-up, DPN-98 [6] ($61.6M$), which indicates that the hierarchical and compositional integration of the DenseNet- and ResNet-aggregation in our AOG building block is more effective than the cascade-based integration in the DPN [6]. Our AOGNet-60M achieves the best results. The FLOPs of our AOGNet-60M are slightly higher than DPN-98 partially because DPN uses ResNeXt operation (i.e., group conv.).

*Model Interpretability* has been recognized as a critical concern in developing deep learning based AI systems [7]. We use the network dissection metric [3] which compares the number of unique "detectors" (i.e., filter kernels) in the last convolution layer. Our AOGNet obtains the best score in comparison (Fig. 6), which indicates the AOG building block has great potential to induce model interpretabilty by design, while achieving the best accuracy performance.

*Adversarial robustness* is another crucial issue faced by many DNNs [2]. We conduct a simple experiment to compare the out-of-the-box adversarial robustness of different DNNs. Table 2 shows the results. Under the vanilla settings, our AOGNets show better potential in adversarial defense, especially when the perturbation energy is controlled relatively low (i.e. $\epsilon = 0.1$). We will investigate this with different attacks and adversarial training in future work.

*Mobile settings.* We train an AOGNet-4M under the typical mobile settings [23]. Table 3 shows the comparison

| Method | #Params | FLOPS | top-1 | top-5 |
|---|---|---|---|---|
| MobileNetV1 [23] | 4.2M | 575M | 29.4 | 10.5 |
| SqueezeNext [14] | 4.4M | - | 30.92 | 10.6 |
| ShuffleNet (1.5) [69] | 3.4M | 292M | 28.5 | - |
| ShuffleNet (x2) [69] | 5.4M | 524M | 26.3 | - |
| CondenseNet (G=C=4) [25] | 4.8M | 529M | 26.2 | 8.3 |
| MobileNetV2 [51] | 3.4M | 300M | 28.0 | 9.0 |
| MobileNetV2 (1.4) [51] | 6.9M | 585M | 25.3 | 7.5 |
| NASNet-C (N=3) [72] | 4.9M | 558M | 27.5 | 9.0 |
| AOGNet-4M | 4.2M | 557M | 26.2 | 8.24 |

Table 3. The top-1 and top-5 error rates (%) on the ImageNet-1K validation set under mobile settings.

results. We obtain performance on par to the popular networks specifically designed for mobile platforms such as the MobileNets [23, 51] and ShuffleNets [69]. Our AOGNet also outperforms the auto-searched network, NASNet [72] (which used around 800 GPUs in search). *We note that we use the same AOGNet structure, thus showing promising device-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.

### 4.3. Object Detection and Segmentation in COCO

MS-COCO is one the widely used benchmarks for object detection and segmentation [40]. It consists of 80 object categories. We train AOGNet in the COCO `train2017` set and evaluate in the COCO `val2017` set. We report the standard COCO metrics of Average Precision (AP), $AP_{50}$, and $AP_{75}$, for bounding box detection ($AP^{bb}$) and instance segmentation, i.e. mask prediction ($AP^m$). We experiment on the Mask-RCNN system [20] using the state-of-the-art implementation, `maskrcnn-benchmark` [42]. We use AOGNets pretrained on ImageNet-1K as the backbones. In fine-tuning for object detection and segmentation, we freeze all the BN parameters as done for the ResNet [21] and ResNeXt [63] backbones. We keep all remaining aspects unchanged. We test both the `C4` and `FPN` settings.

**Results.** Table 4 shows the comparison results. Our AOGNets obtain better results than the ResNet [21] and ResNeXt [63] backbones with smaller model sizes and similar or slightly better inference time. The results show the effectiveness of our AOGNets learning better features in object detection and segmentation tasks.

### 4.4. Ablation Study

We conduct an ablation study which investigates the effects of *(i) RS:* Removing Symmetric child nodes of OR-nodes in the pruned AOG building blocks, and of *(ii) LC:* adding Lateral Connections. 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.

| Method | #Params | $t$ (s/img) | $AP^{bb}$ | $AP^{bb}_{50}$ | $AP^{bb}_{75}$ | $AP^m$ | $AP^m_{50}$ | $AP^m_{75}$ |
|---|---|---|---|---|---|---|---|---|
| ResNet-50-C4 | 35.9M | 0.130 | 35.6 | 56.1 | 38.3 | 31.5 | 52.7 | 33.4 |
| ResNet-101-C4 | 54.9M | 0.180 | 39.2 | 59.3 | 42.2 | 33.8 | 55.6 | 36.0 |
| AOGNet-12M-C4 | 14.6M | 0.092 | 36.8 | 56.3 | 39.8 | 32.0 | 52.9 | 33.7 |
| AOGNet-40M-C4 | 48.1M | 0.184 | **41.4** | **61.4** | **45.2** | **35.5** | **57.8** | **37.7** |
| ResNet-50-FPN | 44.3M | 0.125 | 37.8 | 59.2 | 41.1 | 34.2 | 56.0 | 36.3 |
| ResNet-101-FPN | 63.3M | 0.145 | 40.1 | 61.7 | 44.0 | 36.1 | 58.1 | 38.3 |
| ResNeXt-101-FPN | 107.4M | 0.202 | 42.2 | 63.9 | 46.1 | 37.8 | 60.5 | 40.2 |
| AOGNet-12M-FPN | 31.2M | 0.122 | 38.0 | 59.8 | 41.3 | 34.6 | 56.6 | 36.4 |
| AOGNet-40M-FPN | 59.4M | 0.147 | 41.8 | 63.9 | 45.7 | 37.6 | 60.3 | 40.1 |
| AOGNet-60M-FPN | 78.9M | 0.171 | **42.5** | **64.4** | **46.7** | **37.9** | **60.9** | **40.3** |

Table 4. Mask-RCNN results on *coco_val2017* using the 1x training schedule. Results of ResNets and ResNeXts are reported by the maskrcnn-benchmark.

| 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.

## 5. Conclusions and Discussions

This paper proposes grammar-guided network generators which construct compositional grammatical architectures for deep learning in an effective way. It presents deep AND-OR Grammar networks (AOGNets). The AOG comprises a phrase structure grammar and a dependency grammar. An AOGNet consists of a number of stages each of which comprises a number of AOG building blocks. Our AOG building block harnesses the best of grammar models and DNNs for deep learning. AOGNet obtains state-of-the-art performance. In ImageNet-1K [49], AOGNet obtains better performance than all state-of-the-art networks under fair comparisons. AOGNet also obtains the best model interpretability score using network dissection [3]. AOGNet further shows better potential in adversarial defense. In MS-COCO [40], AOGNet obtains better performance than the ResNet and ResNeXt backbones in Mask R-CNN [20].

**Discussions.** We hope this paper encourages further exploration in learning grammar-guided network generators. The AOG can be easily extended to adopt $k$-branch splitting rules with $k > 2$. Other types of edges can also be easily introduced in the AOG such as dense lateral connections and top-down connections. Node operations can also be extended to exploit grammar-guided transformation. And, better parameter initialization methods need to be studied for the AOG structure.

## Ackowledgement

# References

[1] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. In *Proceedings of the 31th International Conference on Machine Learning, ICML*, pages 584–592, 2014. 2, 3

[2] Anish Athalye and Ilya Sutskever. Synthesizing robust adversarial examples. *CoRR*, abs/1707.07397, 2017. 7

[3] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 7, 8

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

[5] Tianshui Chen, Riquan Chen, Lin Nie, Xiaonan Luo, Xiaobai Liu, and Liang Lin. Neural task planning with AND-OR graph representations. *IEEE Trans. Multimedia*, 21(4):1022–1034, 2019. 4

[6] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. *arXiv preprint arXiv:1707.01629*, 2017. 1, 2, 3, 4, 7

[7] DARPA. Explainable artificial intelligence (xai) program, http://www.darpa.mil/program/explainable-artificial-intelligence, full solicitation at http://www.darpa.mil/attachments/ darpa-baa-16-53.pdf. 7

[8] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *CoRR*, abs/1808.05377, 2018. 4

[9] Pedro F. Felzenszwalb. Object detection grammars. In *IEEE International Conference on Computer Vision Workshops, ICCV*, page 691, 2011. 2, 4

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

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

[12] Stuart Geman, Daniel Potter, and Zhi Yi Chi. Composition systems. *Quarterly of Applied Mathematics*, 60(4):707–736, 2002. 2, 3, 5

[13] 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

[14] Amir Gholami, Kiseok Kwon, Bichen Wu, Zizheng Tai, Xiangyu Yue, Peter Jin, Sicheng Zhao, and Kurt Keutzer. Squeezenext: Hardware-aware neural network design. *arXiv preprint arXiv:1803.10615*, 2018. 8

[15] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015. 7

[16] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017. 7

[17] Ulf Grenander and Michael Miller. *Pattern Theory: From Representation to Inference*. Oxford University Press. 2

[18] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. *IEEE CVPR*, 2017. 3

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

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2980–2988, 2017. 1, 8

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian 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

[22] Geoffrey 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, 2

[23] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 7, 8

[24] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017. 1, 4, 7

[25] Gao Huang, Shichen Liu, Laurens van der Maaten, and Kilian Q Weinberger. Condensenet: An efficient densenet using learned group convolutions. *group*, 3(12):11, 2017. 8

[26] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 3, 4, 6, 7

[27] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian 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

[28] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 448–456. JMLR Workshop and Conference Proceedings, 2015. 2, 4

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

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems (NIPS)*, pages 1106–1114, 2012. 1, 2, 4, 6, 7

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

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

[33] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *CoRR*, abs/1604.00289, 2016. 4

[34] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *CoRR*, abs/1605.07648, 2016. 4

[35] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1, 4

[36] Xiaodan Liang, Liang Lin, and Liangliang Cao. Learning latent spatio-temporal compositional model for human action recognition. *CoRR*, abs/1502.00258, 2015. 4

[37] Liang Lin, Xiaolong Wang, Wei Yang, and Jian-Huang Lai. Discriminatively trained and-or graph models for object shape detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(5):959–972, 2015. 4

[38] Liang Lin, Tianfu Wu, Jake Porway, and Zijian Xu. A stochastic graph grammar for compositional object representation and recognition. *Pattern Recognition*, 42(7):1297–1307, 2009. 4

[39] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013. 3, 4

[40] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. 6, 8

[41] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016. 7

[42] Francisco Massa and Ross Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark, 2018. Accessed: [Insert date here]. 8

[43] David Mumford. Grammar isn't merely part of language. http://www.dam.brown.edu/people/mumford/blog/2016/grammar.html. 2

[44] D. Mumford and A. Desolneux. *Pattern Theory, the Stochastic Analysis of Real World Signals*. AKPeters/CRC Press. 2

[45] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. *CoRR*, abs/1603.06937, 2016. 4

[46] Jongchan Park, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Bam: bottleneck attention module. *arXiv preprint arXiv:1807.06514*, 2018. 7

[47] Geoff Pleiss, Danlu Chen, Gao Huang, Tongcheng Li, Laurens van der Maaten, and Kilian Q. Weinberger. Memory-efficient implementation of densenets. *CoRR*, abs/1707.06990, 2017. 7

[48] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017. 7

[49] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vision (IJCV)*, 115(3):211–252, 2015. 4, 6, 8

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

[51] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 8

[52] Xi Song, Tianfu Wu, Yunde Jia, and Song-Chun 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, 2, 3, 4, 5

[53] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015. 3, 4

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

[55] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. 1, 2, 3, 4

[56] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew 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

[57] Wei Tang, Pei Yu, and Ying 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

[58] Wei Tang, Pei Yu, Jiahuan Zhou, and Ying Wu. Towards a unified compositional model for visual pattern modeling. In *ICCV*, pages 2803–2812. IEEE Computer Society, 2017. 4

[59] John Towns, Timothy Cockerill, Maytal Dahan, Ian T. Foster, Kelly P. Gaither, Andrew S. Grimshaw, Victor Hazlewood, Scott Lathrop, David Lifka, Gregory D. Peterson, Ralph Roskies, J. Ray Scott, and Nancy Wilkins-Diehr. XSEDE: accelerating scientific discovery. *Computing in Science and Engineering*, 16(5):62–74, 2014. 8

[60] Jingdong Wang, Zhen Wei, Ting Zhang, and Wenjun Zeng. Deeply-fused nets. *CoRR*, abs/1605.07716, 2016. 4

[61] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018. 7

[62] Tianfu Wu, Yang Lu, and Song-Chun 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

[63] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for

deep neural networks. *CoRR*, abs/1611.05431, 2016. 1, 2, 3, 4, 7, 8

[64] Quanming Yao, Mengshuo Wang, Hugo Jair Escalante, Isabelle Guyon, Yi-Qi Hu, Yu-Feng Li, Wei-Wei Tu, Qiang Yang, and Yang Yu. Taking human out of learning applications: A survey on automated machine learning. *CoRR*, abs/1810.13306, 2018. 4

[65] Fisher Yu, Dequan Wang, and Trevor Darrell. Deep layer aggregation. In *CVPR*, 2018. 3, 4

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

[67] Ting Zhang, Guo-Jun Qi, Bin Xiao, and Jingdong Wang. Interleaved group convolutions. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 4383–4392, 2017. 3, 4

[68] Xingcheng Zhang, Zhizhong Li, Chen Change Loy, and Dahua Lin. Polynet: A pursuit of structural diversity in very deep networks. *CoRR*, abs/1611.05725, 2016. 4

[69] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017. 8

[70] Long Zhu, Yuanhao Chen, Yifei Lu, Chenxi Lin, and Alan 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, 2, 3, 4, 5

[71] Song Chun Zhu and David Mumford. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, 2(4):259–362, 2006. 1, 2, 3, 4, 5

[72] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2(6), 2017. 8