Towards a Unified Compositional Model for Visual Pattern Modeling

Wei Tang, Pei Yu, Jiahuan Zhou and Ying Wu
Northwestern University
2145 Sheridan Road, Evanston, IL 60208
{wtt450, pyi980, jzt011, yingwu}@eecs.northwestern.edu

Abstract

Compositional models represent visual patterns as hierarchies of meaningful and reusable parts. They are attractive to vision modeling due to their ability to decompose complex patterns into simpler ones and resolve the low-level ambiguities in high-level image interpretations. However, current compositional models separate structure and part discovery from parameter estimation, which generally leads to suboptimal learning and fitting of the model. Moreover, the commonly adopted latent structural learning is not scalable for deep architectures. To address these difficult issues for compositional models, this paper quests for a unified framework for compositional pattern modeling, inference and learning. Represented by And-Or graphs (AOGs), it jointly models the compositional structure, parts, features, and composition/sub-configuration relationships. We show that the inference algorithm of the proposed framework is equivalent to a feed-forward network. Thus, all the parameters can be learned efficiently via the highly-scalable back-propagation (BP) in an end-to-end fashion. We validate the model via the task of handwritten digit recognition. By visualizing the processes of bottom-up composition and top-down parsing, we show that our model is fully interpretable, being able to learn the hierarchical compositions from visual primitives to visual patterns at increasingly higher levels. We apply this new compositional model to natural scene character recognition and generic object detection. Experimental results have demonstrated its effectiveness.

1. Introduction

Compositionality [9, 15] refers to the evident capability of humans to represent entities as hierarchies of parts, which themselves are meaningful and reusable entities. It is believed to be fundamental to all of cognition [7, 1]. For human vision, compositionality enables a unique high-level interpretation of most natural images despite the existence of abundant low-level ambiguities. Besides, compared with the whole objects, potentially with enormous variations, their parts and subparts are less complex. Thus, it is attractive to incorporate compositionality in vision modeling.

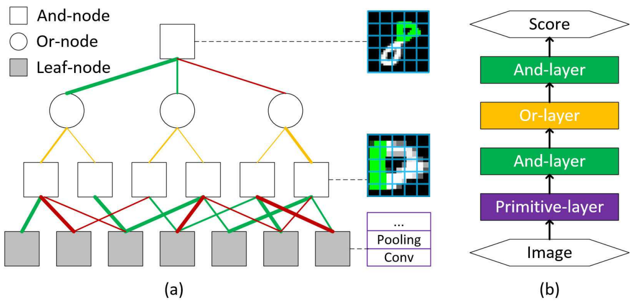

Figure 1. (a) Our approach, based on AOGs, models the compositional structure, parts, features and composition/sub-configuration relationships in a unified framework. The And-node characterizes subpart-part compositions in a local window and involves longer-range contexts via multiscale modeling. Or-nodes point to switchable sub-configurations with different biases (indicated by the line widths). Leaf-nodes model primitives, the lowest-level parts, via CNNs. The structure is discovered via learning the *connection polarities and strengths* (detailed in Sec. 3.3) between And-nodes and their children, respectively indicated by the line colors and widths. (b) The inference is equivalent to a feed-forward network. All the parameters in (a) can be learned end-to-end via BP.

Compositional models[1] aim at modeling the compositionality of patterns. They have been studied in several lines of vision research [44, 15, 43] and exploited in tasks like semantic segmentation [35], object detection [42, 37, 41, 29], and image parsing [44, 24]. Jin and Geman [15] propose a *composition machine* based on a Bayesian network for constructing probabilistic hierarchical image models, designed to accommodate arbitrary contextual relationships. Zhu and Mumford [44] quest for a stochastic and context sensitive grammar of images, embodied in a simple And-Or graph (AOG), to model a large number of object categories. Zhu *et al*. [43] also exploit the AOG to design the recursive compositional model but put more emphasis on discriminative techniques and efficient inference and learning algorithms.

However, there are problems with all the existing com-

[1] We focus on multilevel compositional models in this paper.

positional models. First, many previous approaches manually design the compositional architectures and require the annotations of objects and parts at all levels [44, 15, 9]. This kind of annotations are difficult to obtain. Moreover, the predefined parts and architectures might be suboptimal and result in inferior training [6]. Second, when only image/object-level labeling is available, the compositional structures and part placements are usually mined from the data in a bottom-up fashion [42, 43, 35, 37, 40]. The separation of structure and part discovery from parameter learning generally leads to suboptimal learning of the model. Third, due to the lack of part annotations, latent structural learning is exploited for parameter estimation [43, 41, 35, 38, 24]. It is difficult to scale up to big data for deep structures [41].

To address these difficult issues, we propose a novel compositional model, termed as *composition network* (CompNet), for image pattern modeling. As illustrated in Fig. 1, it is represented by an AOG. Given a dataset of visual patterns with only image/object-level labels, our approach can learn the structure, parts, features and composition/sub-configuration relationships in a unified framework. Then, it can be used for pattern classification or detection. Specifically, we propose a multiscale nonparametric model to characterize the part-subpart relationships. It not only makes CompNets more flexible but also simplifies the part discovery. Instead of adopting predefined primitives and hand-crafted features, we utilize the convolutional neural network (CNN) [18] to model them. Hence, CompNets can be more adaptable. By learning both the valid and invalid components of a part/object as well as their importance to the task, the structure can be discovered simultaneously with parameter estimation. The consideration of irrelevant components also makes CompNets discriminative. Finally, we show the inference algorithm of a CompNet is equivalent to a feed-forward network (FFN). Thus, all the parameters can be learned end-to-end via the highly-scalable BP [26].

In summary, the novelty of this paper is as follows:

- To the best of our knowledge, the proposed CompNet is the first of its kind to unify the following key ingredients in compositional modeling: structure, parts, features and composition/sub-configuration relations. The unification makes it flexible and adaptable.

- This is also the first attempt to relate an AOG to an FFN and combine it with CNNs. As a result, all the parameters of CompNets can be learned end-to-end via BP. This not only makes the learning scalable to big data but also enables a better fitting of the model.

- Compared with artificial neural networks, *e.g.*, CNNs, our model is fully interpretable. We show in the experiments that the learned primitives can be composed recursively to form increasingly higher-level patterns. Besides, top-down parsing of an input pattern enables us to trace its part locations and choices of sub-configurations at each level.

2. Related Work

Compositional models. As described in the previous section, existing compositional models [9, 44, 15, 42, 43, 35, 41, 37, 38, 24, 40] treat structure discovery, part modeling and parameter estimation seperately. Our approach attempts to address these problems in a unified framework, leading to a more flexible, adaptable and scalable compositional model.

Convolutional neural networks (CNNs). Our model is inherently different with CNNs. The CompNet is based on a probabilistic graphical model defined on an AOG, with explicit composition/sub-configuration semantics embedded in each node. Thus, it is fully interpretable. In contrast, CNNs are compositions of nonlinear functions without clear semantics. As pointed out by Fodor and Pylyshyn [7], neural networks fail to mimic the basic compositionality of human cognition. Instead, they mean to simulate the responses of individual neurons to stimuli within their receptive fields. Though meaningful object parts have been observed at the intermediate layers [39], their relationships remain unclear. In sum, our CompNets aim to model compositional structures of patterns, while CNNs are to learn powerful features. Thus, it is natural to integrate them.

CNN-DPM. Recently, there have been attempts [32, 12, 34] to formulate graphical models as FFNs and attach CNNs before them as feature extractors for end-to-end training. The work most related to ours is the CNN-DPM frameworks [12, 34], where deformable part-based models (DPMs) [6], unrolled as FFNs, are combined with CNNs to model the object-part relationships. Ouyang *et al*. [23] also generalize DPMs as pooling layers and insert them between successive convolutional layers to constrain the geometric distributions of the neural activations. Our approach differs from them in that: i) it exploits recursive composition/sub-configuration relationships between patterns at different levels, ii) it learns the compositional structure by considering both relevant and irrelevant components of a pattern as well as their importance, and iii) it learns subpart-part displacements automatically and shows how to use them for bottom-up composition from primitives to patterns at increasingly higher levels.

Sprite models. Unlike our decomposing object patterns into their parts and subparts and focusing on discriminative learning, sprite models [16, 36, 2] represent image scenes as layered compositions of flexible sprites and backgrounds via probabilistic generative models. Unsupervised learning methods such as the EM algorithm [4] are used for parameter estimation and object discovery.

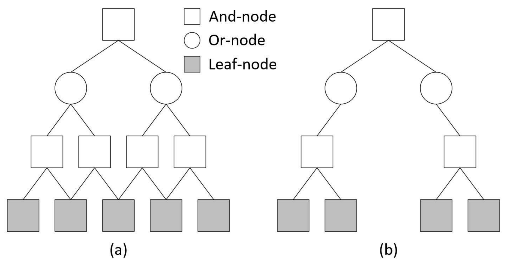

Figure 2. (a) An example AOG with a predefined architecture. (b) A parse graph of the AOG in (a).

3. Composition network

A composition network (CompNet) is a compositional model that takes as input an image and produces as output scores for each pattern class. Represented by an AOG, it models the compositional structure, features, parts and composition/sub-configuration relationships in a unified framework. Its inference process is equivalent to an FFN, whose parameters can be learned efficiently by BP.

3.1. And-Or graph

An And-Or graph (AOG), as shown in Fig. 2 (a), is defined by a 5-tuple $(\mathcal{V}, \mathcal{E}, \phi^{and}, \phi^{or}, \phi^{leaf})$, which specifies its graph structure $(\mathcal{V}, \mathcal{E})$ and potential functions $(\phi^{and}, \phi^{or}, \phi^{leaf})$.

An AOG can have three types of nodes: $\mathcal{V} = \mathcal{V}^{and} \cup \mathcal{V}^{or} \cup \mathcal{V}^{leaf}$. And-nodes $\mathcal{V}^{and}$ model the composition of components into a higher-level part/object. Or-nodes $\mathcal{V}^{or}$ model the switches among sub-configurations of a concept, *e.g.*, different views or subcategories. Leaf-nodes $\mathcal{V}^{leaf}$, having no child, model the primitives, the lowest-level parts that cannot be decomposed further. We only consider vertical edges in this paper. Let $ch(u)$ denote the set of children of a node u. An edge connects an And/Or-node u with its child $v \in ch(u)$: $\mathcal{E} = \{(u, v) : u \in \mathcal{V}^{and} \cup \mathcal{V}^{or}, v \in ch(u)\}$. Note different And-nodes at the same level can share their children but Or-nodes cannot. We call the And/Or-nodes at the highest level as root nodes. For multi-class pattern modeling, we can always use an Or-node as the root node, whose children represent all the target objects. We call this child level as an Object-level.

A parse graph is a subgraph of an AOG derived by making choices at each Or-node. It represents a hierarchical interpretation of a specific image. Since each Or-node can point to different children, an AOG can have multiple valid parse graphs and correspondingly, an image may have more than one valid (but not necessarily good) interpretation. An example parse graph is shown in Fig. 2 (b).

State variables are associated with each node $u \in \mathcal{V}$. For And-nodes and Leaf-nodes, the state variable w_u represents the spatial location of a part/primitive. Each Or-node has two types of state variables. z_u denotes the switch among its children: $z_u \in ch(u)$. w_u stores the location of the selected child: $w_u \equiv w_{z_u}$. Let Ω denote the set of all state variables in the AOG: $\Omega = \{w_u : u \in \mathcal{V}^{and} \cup \mathcal{V}^{leaf}\} \cup \{(w_u, z_u) : u \in \mathcal{V}^{or}\}$. Note a valid Ω should always correspond to a parse graph. For those nodes not included in a parse graph, we define their state variables as empty: $w_u = \emptyset$ or $z_u = \emptyset$ and their potential functions as zero: $\phi = 0$.

The probability distribution over the state variables Ω is of the following Gibbs form:

$$p(\Omega|\mathbf{I}) = \frac{1}{Z} \exp\{-E(\Omega, \mathbf{I})\} \tag{1}$$

where $\mathbf{I}$ is the input image, $E(\Omega, \mathbf{I})$ is the energy and Z is the partition function. For convenience, we use a score function, defined as the negative energy, to specify the model and omit $\mathbf{I}$:

$$S(\Omega) \equiv -E(\Omega, \mathbf{I}) = \sum_{u \in \mathcal{V}^{leaf}} \phi_u^{leaf}(w_u, \mathbf{I}) + \sum_{u \in \mathcal{V}^{and}} \sum_{v \in ch(u)} \phi_{u,v}^{and}(w_u, w_v) + \sum_{u \in \mathcal{V}^{or}} \phi_u^{or}(z_u, w_u) \tag{2}$$

where the three terms are potential functions corresponding to Leaf, And and Or nodes, respectively. The first term acts like a detector: it determines how likely the primitive modeled by the Leaf-node u is present at w_u in the image. The second term models the preferred position of the child v relative to its parent u. The last term models the Or-nodes' selection biases towards each of their children.

In Eq. (2), only Leaf-nodes are related to the observed images. By including extra data potentials, we can also associate the higher-level And-nodes to the observations. To put more emphasis on the relational modeling, we derive our framework using Eq. (2) while the derivation of the extended model is straightforward and similar.

3.2. Node models

We first decouple the three node models by computing $S(\Omega)$ in a recursive and level-by-level fashion. Let $\mathcal{V}_u \subseteq \mathcal{V}$ denote the set of node u and all its descendants. Note $\mathcal{V}_u$ and the corresponding edges form a new AOG, which is a subgraph of the AOG $(\mathcal{V}, \mathcal{E})$. We use $\Omega_u \subseteq \Omega$ to represent the set of state variables of the nodes in $\mathcal{V}_u$. Thanks to the hierarchical structure of AOGs, the score $S(\Omega)$ can be computed recursively using the following three equations:

$$\text{(And)}\ S_u(\Omega_u) = \sum_{v \in ch(u)} \phi_{u,v}^{and}(w_u, w_v) + S_v(\Omega_v) \tag{3}$$

$$\text{(Or)}\quad S_u(\Omega_u) = \phi_u^{or}(z_u, w_u) + S_{z_u}(\Omega_{z_u}) \tag{4}$$

$$\text{(Leaf)}\ S_u(\Omega_u) = S_u(w_u) = \phi_u^{leaf}(w_u, \mathbf{I}) \tag{5}$$

where $S_u(\Omega_u)$ is the score of the AOG rooted at u and defined similarly as Eq. (2). The recursion begins from the

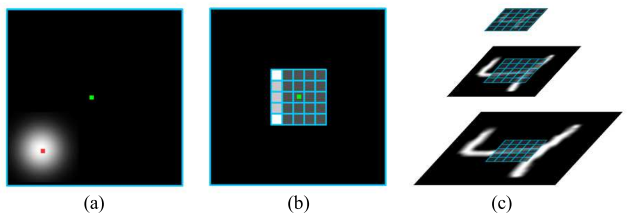
Figure 3. (a) Previous compositional models use an anchor point (the red pixel) and Gaussian distributions to model the displacement of a child from its parent's position (the green pixel). (b) Our And-node characterizes the composition relationship in a local window. Each position is assigned a bias indicating the prior the child is placed there. (c) By modeling the higher-level part's position in lower resolutions, the local model can involve increasingly larger image regions.

Leaf-nodes and is initialized by Eq. (5). Then, Eq. (3) or Eq. (4) will be called whenever an And-level or an Or-level is met until the root node is reached. We refer to these three equations as node models of an AOG.

3.2.1 And-node

In existing works, the And-node is always modeled parametrically as: $\phi^{and}_{u,v}(w_u, w_v) = \lambda^{and}_{u,v} \cdot h_{u,v}(w_u, w_v)$, where $\lambda^{and}_{u,v}$ and $h_{u,v}(w_u, w_v)$ represent the parameters and features, respectively. Fig. 3(a) shows a typical example: the feature function $h_{u,v}(w_u, w_v)$ exploited in [43, 35, 41, 29, 37] is quadratic in the displacement of a child node from its reference position, and hence corresponds to Gaussian distributions. Such kind of parametric models result in some potential problems. i) There may not exist a single ideal anchor point for some parts. For example, the head of a horse is high in the air when it is walking but close to the ground when it is eating or drinking. ii) Even with such unique reference locations, the distributions of the parts' positions may not be Gaussian. They may not even be symmetric. iii) It introduces a critical parameter, *i.e.*, the anchor position of a part relative to its parent. It is obtained via manual annotation [15, 41] or bottom-up clustering [43, 35], both of which may be suboptimal and lead to inferior learning.

To attack these problems, we propose a multiscale nonparametric model for the And-node. As shown in Fig. 3(b), we use a local window $\mathbb{D}_v = [-c, c] \times [-c, c]$ to characterize the patterns of the child's location relative to its parent. Each position is assigned a bias to indicate the prior the child is placed there. Obviously, this model is able to represent any local subpart-part displacement and avoids the specification of anchor points. For example, Fig. 3(b) suggests that the child is most likely present at the left part of its parent. The longer-range context is modeled via the hierarchical structure of the CompNet and multiscale modeling, as shown in Fig. 3(c). Using a lower resolution for the parent's positions can enlarge the context an And-node can model on the image space. If we periodically insert downsampling between successive And-nodes in the architecture, the higher-level nodes will model larger context. This agrees with our intuition that the child-parent displacements of lower-level compositions, *e.g.*, from edges to their junctions, should be smaller than those of higher-level ones, *e.g.*, from semantic parts to objects. Specifically, Eq. (3) is instantiated as

$$S_u(\Omega_u) = \sum_{v \in ch(u)} \lambda^{and}_{u,v}(\bar{w}_v) + S_v(\Omega_v) \tag{6}$$

where $\bar{w}_v \equiv w_v - w_u$ ($\bar{w}_v \in \mathbb{D}_v$) denotes the displacement of the child v w.r.t. the parent u. $\lambda^{and}_{u,v}()$ is a lookup table mapping $\bar{w}_v \in \mathbb{D}_v$ to a real value. It should account for the natural frequency a child appears somewhere relative to its parent. Here, we implicitly assume the child node has been scaled to the same resolution as its parent.

3.2.2 Or-node

The potential function of an Or-node $\phi^{or}_u(z_u, w_u)$ reflects the priors of selecting each child. We can simply assign each $z_u \in ch(u)$ a real value $\lambda^{or}_u(z_u)$ and Eq. (4) becomes:

$$S_u(\Omega_u) = \lambda^{or}_u(z_u)\delta(w_u = w_{z_u}) + S_{z_u}(\Omega_{z_u}) \tag{7}$$

where $\lambda^{or}_u()$ are parameters and account for the natural frequency of each child, $\delta(w_u = w_{z_u})$ constrains that the location of the selected child will be transmitted to the Or-node.

3.2.3 Leaf-node

Leaf-nodes model the primitives. Acting as the cornerstone for the composition of all the higher-level parts/objects, they should be modeled with two principles: reusable and discriminative. The first principle drives us to learn primitives from the training images. To best fit the data, primitives that occur most will pop up. Compared with some predefined primitives, such as rectangles and curves [44, 29, 38], learned primitives would be more adaptable. Recently, deep features [14] have been proved to outperform handcrafted ones by a large margin in several pattern recognition tasks. This motivates us to exploit CNNs to extract discriminative features. Specifically, the Leaf-node is modeled as

$$S_u(\Omega_u) = \lambda^{leaf}_u \cdot f_u(w_u, \mathbf{I}; \Theta) \tag{8}$$

where λ^{leaf}_u is the primitive filter; $f_u(w_u, \mathbf{I}; \Theta)$ is CNN features extracted at location w_u of image $\mathbf{I}$ with Θ being the collection of the CNN's parameters. Both λ^{leaf}_u and Θ will be learned from the data. Note Eq. (8) is general and can be used as data potentials for the higher-level And-nodes.

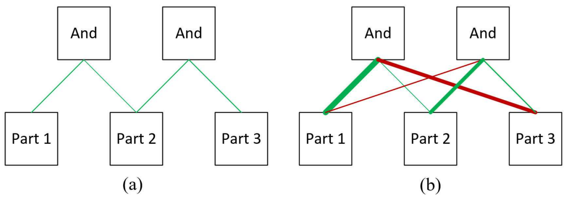

Figure 4. (a) Existing compositional models only connect each And-node to relevant parts and assume they contribute equally to the composition. (b) An And-node in CompNets not only considers both relevant (green) and irrelevant (red) components but also assigns them with different importance (indicated by line widths).

3.3. Structure

Till now, we have assumed a given and fixed architecture of an AOG, *e.g.*, Fig. 2(a). It is often defined manually or discovered in a bottom-up fashion. In this paper, we propose to unify the structure discovery and parameter estimation. We assume the level number in the AOG, the level types and the node number in each level are known. Since the Or-nodes at the same level do not share child nodes, we simply assign them with the same number of children.

For the And-nodes, we first connect each of them with all the nodes at one level lower and then model the structure via a set of *connection parameters*, whose signs and magnitudes indicate the *connection polarities and strengths*, respectively. Assume $l(u)$ returns the index of the level node u belongs to. $\mathcal{V}^{l(u)}$ denotes the set of nodes on level $l(u)$. Then, the And-node model in Eq. (6) is reformulated as:

$$S_u(\Omega_u) = \sum_{v \in \mathcal{V}^{l(u)-1}} \lambda_{u,v}^{and}(\bar{w}_v) + \lambda_{u,v}^{con} S_v(\Omega_v) \tag{9}$$

where $\lambda_{u,v}^{con} \in \mathbb{R}$ is the connection parameter between nodes u and v. Note multiplying $\lambda_{u,v}^{con}$ to the first term is redundant because it will be absorbed. The sign of $\lambda_{u,v}^{con}$ indicates the connection polarity: whether node v is a relevant component of node u. A positive value of $\lambda_{u,v}^{con}$, which means v is likely to be a part of u, makes S_u a monotonically increasing function of S_v: seeing v increases the confidence that u appears. A more interesting case is when $\lambda_{u,v}^{con} < 0$. Then, a larger S_v leads to a smaller S_u. It means that when v is not a component of u, seeing v will decrease the probability that u appears, which is consistent with our intuition. As pointed out by Fukushima [8], checking the absence of irrelevant features plays an important role in pattern recognition. Thus, the consideration of irrelevant components will make CompNets more discriminative. The magnitude of $\lambda_{u,v}^{con}$ indicates the connection strength between nodes u and v. It evaluates the importance of this connection to the target task and allows CompNets to put more emphasis on the discriminative parts in the composition. Fig. 4 compares the structure modeling of CompNets and previous methods.

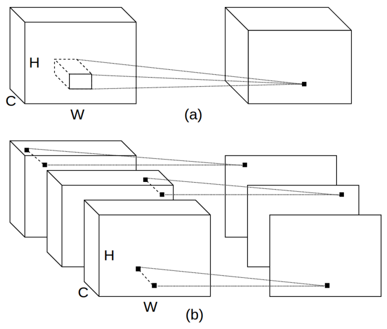

Figure 5. The connection between the input and output feature maps of (a) an And-layer and (b) an Or-layer. Each cube denotes a set of stacked feature maps, whose channel and spatial dimensions are annotated by C and (H, W), respectively.

3.4. Inference as a feed-forward network

Since each child node may have multiple parents, there will be closed loops in our AOG. Empirical studies [42, 43] show that the update rules of dynamic programming, which can be thought of as belief propagation using message passing [25], can obtain good approximation for inference on such kind of AOGs. Thus, the maximum score $S^* = \max_\Omega S(\Omega)$ for an input image $\mathbf{I}$ can be computed recursively with the following three equations:

$$(\text{And})\ S_u^*(w_u) = \sum_{v \in \mathcal{V}^{l(u)-1}} \max_{\bar{w}_v \in \mathbb{D}_v} \lambda_{u,v}^{and}(\bar{w}_v) + \lambda_{u,v}^{con} S_v^*(w_v) \tag{10}$$

$$(\text{Or})\quad S_u^*(w_u) = \max_{v \in ch(u)} \lambda_u^{or}(v) + S_v^*(w_u) \tag{11}$$

$$(\text{Leaf})\ S_u^*(w_u) = S_u(w_u) = \lambda_u^{leaf} \cdot f_u(w_u, \mathbf{I}; \Theta) \tag{12}$$

where $S_u^*(w_u)$ is the maximum score of the subgraph formed by $\mathcal{V}_u$ with root node u taking state w_u, and is computed recursively by Eq. (10) or Eq. (11), depending on the node type, with boundary conditions provided by Eq. (12). The recursion begins from the Leaf-level and goes up until the root node is reached. At the Object-level, we obtain scores for each pattern class. w_v in Eq. (10) is a function of $\bar{w}_v$ and thus should be included in the max operation.

The function $S_u^*(w_u)$, which maps the node index u and 2-D position w_u to real values, can be thought of as $|\mathcal{V}^{l(u)}|$ stacked *feature maps*, whose channel and spatial dimensions are indexed by u and w_u, respectively. Then, Eqs. (10)-(12) can be interpreted as *And, Or and Primitive layers*, transforming input feature maps (S^* or f on the RHS) to the output ones (S^* on the LHS). Thus, the inference of a CompNet is equivalent to an FFN, as shown in Fig. 1(b).

The Primitive-layer defined in Eq. (12) can be considered as a CNN, formed by stacking a convolution layer [18] with weights λ_u^{leaf} and the CNN feature extractor $f_u(w_u, \mathbf{I}; \Theta)$. An And-layer is illustrated in Fig. 5(a). Each output unit (u, w_u) is connected to a local region of the input feature maps $\{(v, w_v) : v \in \mathcal{V}^{l(u)-1}, w_v - w_u \in \mathbb{D}_v\}$. An Or-layer is illustrated in Fig. 5(b). The input feature maps are divided into groups, each of which will output a one-channel feature map with the same spatial dimensions as the input. Indeed, the only difference between the formulations of the Or-layer and the Maxout function [13] is that the former has bias terms. From a Bayesian point of view, when the likelihoods of two sub-configurations are similar, the one that occurs more frequently in nature should be selected. Thus, the bias term is necessary.

Another building block for our CompNet is the max-pooling layer, which has been widely used in CNNs. It is necessary for several reasons. First, it is used for multiscale modeling, discussed in Sec. 3.2.1. Second, since the exact location of a part is less important than its rough displacement relative to its parent, max-pooling will make And-layers robust to small shifts on the image. Third, it reduces the sizes of feature maps and thus the computation.

3.5. Learning via BP

The inference process of a CompNet is an FFN consisting of four types of layers, *i.e.*, And, Or, Primitive and max-pooling, with the Primitive-layer being a CNN. Thus, it is natural to learn all the parameters $(\lambda^{and}, \lambda^{or}, \lambda^{leaf}, \lambda^{con}, \Theta)$ via BP in a unified framework. For multi-class classification, we exploit the Softmax loss [14]. When an input pattern is present to a CompNet, it is propagated forward through the network, layer by layer, until it reaches the Object-level. The scores of each object class are then compared with the label using the loss function. An error value is calculated, which is then propagated backwards, starting from the loss layer, all the way to the Primitive-layer. Partial derivatives of the loss function w.r.t. each parameter can be obtained from the back-propagated errors and used to update these parameters.

We alleviate overfitting in two ways. First, we constrain the model complexity by setting both the size of $\mathbb{D}_v$ and the number of each Or-node's children small. Second, inspired by the recent success in training deep CNNs [14], we exploit large training datasets, data augmentation, pre-trained CNN feature extractors and/or dropout techniques [30].

4. Experiments

We apply CompNets to solve three practical problems of increasing complexities. First, the MNIST handwritten digit database [18] is used to demonstrate the CompNet's capability for compositional modeling and its interpretability. Then, we apply a CompNet to natural scene character recognition, a natural extension of the first task. Finally, we show CompNets can be applied to generic object detection.

We implement the CompNet using Caffe [14] and train it end-to-end with stochastic gradient descent (SGD) and BP. $(\lambda^{and}, \lambda^{or})$ are initialized to 0. We exploit a data-driven method [20] to initialize $(\lambda^{leaf}, \lambda^{con}, \Theta)$. A momentum of 0.9 and parameter decay of 0.0005 are used. Since the properties of the datasets, *e.g.*, image sizes and variations, vary greatly from task to task, we use different batch sizes and learning rates for them, which will be detailed later. We adopt a base learning rate [14] and reduce it by a factor of 10 when the loss on the validation set stops decreasing.

4.1. Model validation

The goal of this experiment is to investigate what the CompNet has learned from the data instead of comparing it with the state-of-the-art algorithms. The MNIST dataset consists of 28×28 pixel greyscale images of handwritten digits 0-9, with 60,000 training and 10,000 test examples. Since the patterns in this dataset are not complex, we exploit a relatively simple architecture: data-conv(k5c20)-pool-and(k5c200)-pool-or(c100)-and(k4c10)-loss[2]. No data augmentation is used in this experiment. The batch size and base learning rate are set to 100 and 0.05, respectively. With this architecture, the CompNet obtains a test set error of 0.80%. As a baseline, the LeNet-5 [18] implemented in Caffe achieves a test set error of 0.89%. Its architecture is: data-conv(k5c20)-pool-conv(k5c50)-pool-fc(c500)-relu-fc(c10)-loss. If we treat the Or-layer as an activation function, in analog to the Maxout and ReLU activations, our CompNet is indeed one layer shallower and also narrower than the LeNet-5, but can still get a slightly better result.

4.1.1 Bottom-up composition of filters via recursion

Beginning from the primitive filters, *i.e.*, weights of conv(k5c20), we use the learned parameters to recursively compose filters corresponding to patterns of increasingly higher levels. No input image is required. There are larger numbers of valid filters for higher-level nodes due to i) the local shift-invariance modeled by the pooling layer, ii) sub-configurations modeled by the Or-nodes and iii) potential multimodal distributions of the subpart-part displacements.

We first show one of the most preferred filters for each node to illustrate the composition process. Specifically, it begins from the primitive level and goes up all the way to the Object-level. For an And-node u without pooling before it, its filter is formed by first shifting each child filter v to the learned most preferred location $\bar{w}_v^* = \arg\max_{\bar{w}_v} \lambda_{u,v}^{and}(\bar{w}_v)$ and then linearly combining them with weights $\lambda_{u,v}^{con}$. If pooling exists, the values of $\bar{w}_v^*$ should be doubled and result in 4 possible locations (a 2×2 local region). We simply use the top-left position. An Or-node can switch between two child filters. We choose the child with greater bias.

[2]Pooling layers with kernel size 2×2 and stride 2 are always used; 'k' and 'c' respectively specify the kernel (window) size and channel number for the And or convolution layer, *e.g.*, 'k5c20' means kernel size 5×5 and channel number 20. 'fc' will be used to denote a fully-connected layer.

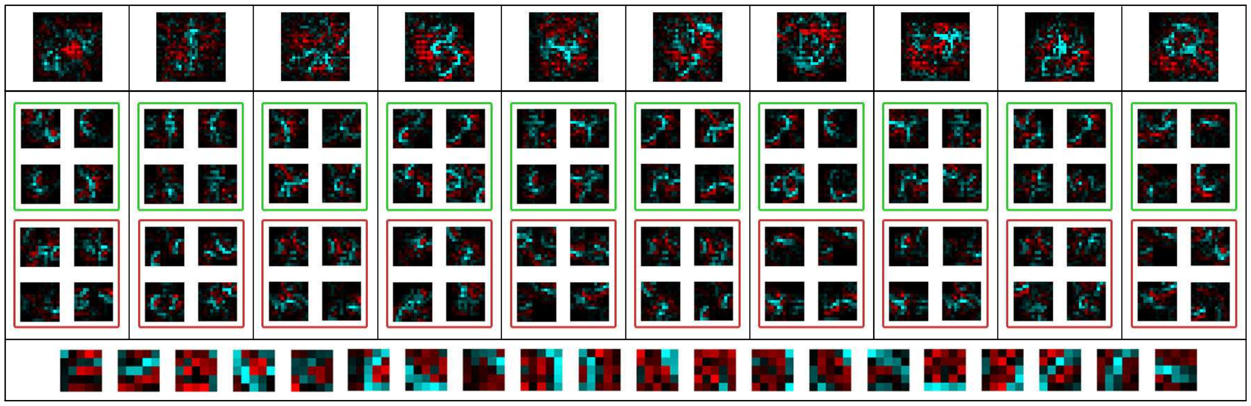
Figure 6. Node filters learned on the MNIST dataset. Cyan and red are used to encode the filters' positive and negative values, with brighter colors indicating larger absolute values. Each higher-level filter is composed from the child filters using the learned parameters. Bottom row: 20 primitive filters. Top row: 10 Object-level filters. Middle row: 80 Or-node filters. The 4 filters in a green (red) box are those most relevant (irrelevant) to the corresponding object filter, evaluated by λ_{uv}^{con}. Due to part sharing, some filters may occur in multiple columns.

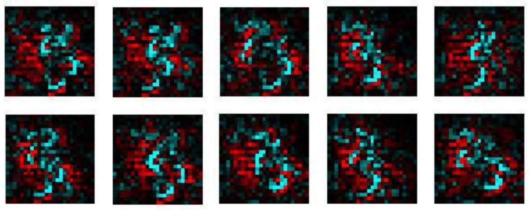
Figure 7. Choosing different sub-configurations in the process of bottom-up composition leads to variations of a pattern filter.

The composed filters are shown in Fig. 6. We can observe that the primitive filters (bottom row) are mostly edge detectors. The shapes of the object filters (top row) clearly correspond to the 10 digits. The brighter parts of a digit are caused by larger magnitudes of $\lambda_{u,v}^{con}$, *i.e.*, the connection strength. As expected, they are related to the discriminative parts of each digit, *e.g.*, the middle part of '4' and upper part of '7'. The Or-node filters (middle row) are joints of edges, representing relevant or irrelevant parts of each digit. We can easily see some part sharing among different columns. As shown in Fig. 7, choosing different sub-configurations at each node in the composition process leads to a large amount of valid filters for higher-level patterns. This is how CompNets differ from simple template matching and is able to model enormous variations of a pattern.

4.1.2 Top-down parsing of objects via backtracking

Given an input image, top-down pass is used to find the best state configurations Ω^* and the corresponding parse graph of an AOG. It recursively inverts Eqs. (10) and (11) to obtain the optimal states of the child nodes that yield the maximum node scores. The complete parse graph, which interprets both the backgrounds and objects, contains a large number of paths from the root node to the Leaf-nodes. We define the score of a path as the multiplication of all the

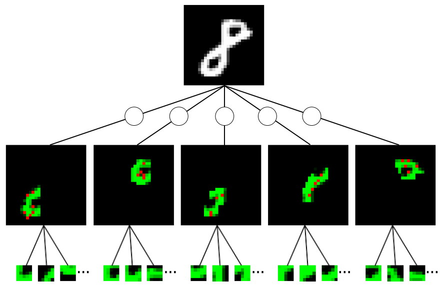
Figure 8. Top-down parsing of an input image (top) to its parts (the third row) and primitives (bottom) by making choices at the Or-nodes (the second row). Red pixels denote the locations of primitives on each part.

node scores on this path. To focus on the parsing of objects and avoid too large parse graphs (so that we could display), we only retain paths with scores larger than 0.3.

Fig. 8 gives an example of the obtained parse graphs. Fig. 9 shows the parsing of input images to their primitives. By placing the corresponding primitive filters on their backtracked locations, we can obtain shapes similar to the input digits. Note that the densities of primitives on each part of a digit can be quite different and roughly correspond to the brightness of these parts on Fig. 6, *e.g.*, the middle part of '4', upper parts of '7' and '9' and lower part of '2'.

4.2. Natural scene character recognition

Three datasets, *i.e.*, ICDAR-03 [19], IIIT5K [21] and Chars74K [3], are used here. Following [27, 31], characters with similar structures, such as 'X' and 'x', 'P' and 'p', 'K' and 'k', are relabeled as the same class, resulting in 49 classes. For the first two datasets, we use their own splitting of the data for training and testing. The Chars74K dataset assigns only 930 out of 7705 samples as training set, which makes our model overfitting. Thus, we use the model trained on the ICDAR-03 training set

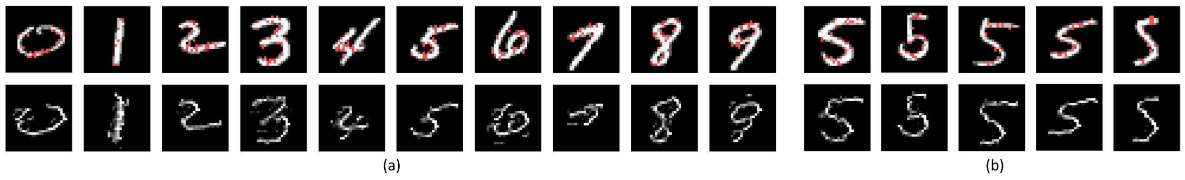

Figure 9. (a) Top-down parsing of input images to their primitives. First row: backtracked primitive locations (red pixels) superposed on the input images. Second row: placing the corresponding primitive filters on these locations leads to shapes similar to the input digits. For clearer visualization, only the positive parts of the composed filters are displayed. (b) Top-down parsing of the same pattern with variations.

	ICDAR-03	IIIT5K	Chars74K
[31]	79.9	-	-
[22][3]	**86.0**	85.0	-
[27]	77.9	-	71.7
CNN	83.0	87.7	83.3
Maxout	82.7	87.9	83.6
Ours	85.3	**88.9**	**85.3**

Table 1. Character classification accuracy (in %) on 3 datasets.

to test on all the Chars74K samples. All the training images are resized to 32×32. Simple data augmentations, *i.e.*, shifting and rotation, are used. The CompNet architecture is: data-conv(k3c32)-conv(k3c32)-pool-and(k5c128)-or(c64)-and(k3c256)-or(c128)-and(k3c49)-loss. The batch size and base learning rate are set to 128 and 0.05, respectively. We compare the CompNet with i) a part-based model [27], ii) state-of-the-art methods using HOG [31] and CNN [22] features, iii) CNN and Maxout networks with similar architectures with our CompNet, *i.e.*, the same layer numbers, kernel sizes and channel numbers. The results are shown in Tab. 1. Note extra datasets, including the Chars74K, are exploited to train the CNN used in [22]. If we also include the Chars74K as training set in the ICDAR-03 task, the CompNet's classification accuracy is 88.1%.

4.3. Object detection

We apply CompNets to the PASCAL VOC 2007 object detection task [5]. It contains 20 object classes. Mean average precision (mAP) [5] is used to evaluate the results. The CNN-DPMs [12, 34] exploit the *conv*5 features of the pre-trained AlexNet [17] or its variant [39]. To have a fair comparison with them, we also use the *conv*5 features of the AlexNet in our CompNet. We follow the region-based CNN (R-CNN) frameworks [11, 10], which generate object proposals [33] from each input image, later refined by a regression, and utilize the region of interest (RoI) pooling layer [11] to extract feature maps of fixed size, *i.e.*, $6 \times 6 \times 256$, for each proposal from the *conv*5 features. Specially, we respectively use and(k4c256)-or(c128)-and(k3c512)-and(k1c21) for structure modeling and fc(c21) as Object-level data potentials to obtain the classification

[3]Extra data are used to train the CNN feature extractor.

method	CNN-DPM [12]	CNN-DPM [34]	RCM [41]
mAP	45.2	46.9	29.6
method	R-CNN p5 [11]	AOT [29]	Ours
mAP	47.3	34.7	**52.0**

Table 2. mAP (in%) on PASCAL VOC 2007. All CNN-based methods use *conv*5 of pre-trained AlexNets as feature extractors.

scores for the 20 object classes and background. For better convergence, the And/Or-layers are first trained using a larger batch size, *i.e.*, 10. Then the whole network is trained as [11, 10] with base learning rate 0.001 and batch size 2. We compare our method with CNN-DPMs [12, 34], two state-of-the-art compositional models designed for this task, *i.e.*, the recursive compositional model (RCM) [41] and the And-Or tree (AOT) [29], and the R-CNN version using *pool5* features (R-CNN p5) [11]. We have tried to train a Fast R-CNN [10] using *pool5* features but fail to make it converge. As shown in Tab. 2, the proposed approach outperforms the baseline methods. The R-CNN with *fc6* and *fc7* layers can achieve a better mAP, *i.e.*, 58.5%. Extending our model to use deeper networks can also provide similar gains from better feature representations. For example, we have tested another version of the CompNet with the pre-trained VGG16 network [28] as feature extractor and obtain 67.4% mAP. This is why we constrain all CNN-based methods to use features from the same network and same depth.

5. Conclusion

This paper presents a novel compositional model for visual pattern modeling. It learns the structure, parts, features and composition/sub-configuration relationships via BP in an end-to-end and unified fashion. As a result, our framework is flexible, adaptable and scalable. The validation experiments demonstrate that the learned compositionality is fully interpretable. We also show it can be applied to natural scene character recognition and generic object detection. Quantitative results demonstrate its effectiveness.

Acknowledgement

This work was supported in part by National Science Foundation grant IIS-1217302, IIS-1619078, and the Army Research Ofice ARO W911NF-16-1-0138.

References

[1] E. Bienenstock and S. Geman. Compositionality in neural systems. *The handbook of brain theory and neural networks*, 1995. 1

[2] Z. Dai and J. Lücke. Unsupervised learning of translation invariant occlusive components. In *CVPR*, 2012. 2

[3] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *VISAPP (2)*, 2009. 7

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, 1977. 2

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 8

[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010. 2

[7] J. A. Fodor and Z. W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 1988. 1, 2

[8] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 1980. 5

[9] S. Geman, D. F. Potter, and Z. Chi. Composition systems. *Quarterly of Applied Mathematics*, 2002. 1, 2

[10] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 8

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 8

[12] R. Girshick, F. Iandola, T. Darrell, and J. Malik. Deformable part models are convolutional neural networks. In *CVPR*, 2015. 2, 8

[13] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Maxout networks. *ICML*, 2013. 6

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 4, 6

[15] Y. Jin and S. Geman. Context and hierarchy in a probabilistic image model. In *CVPR*, 2006. 1, 2, 4

[16] N. Jojic and B. J. Frey. Learning flexible sprites in video layers. In *CVPR*, 2001. 2

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 8

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. 2, 5, 6

[19] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. Icdar 2003 robust reading competitions. In *ICDAR*, 2003. 7

[20] D. Mishkin and J. Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015. 6

[21] A. Mishra, K. Alahari, and C. Jawahar. Scene text recognition using higher order language priors. In *BMVC*, 2012. 7

[22] A. Mishra, K. Alahari, and C. Jawahar. Enhancing energy minimization framework for scene text recognition with top-down cues. *CVIU*, 2016. 8

[23] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, et al. Deepid-net: Deformable deep convolutional neural networks for object detection. In *CVPR*, 2015. 2

[24] S. Park and S.-C. Zhu. Attributed grammars for joint estimation of human attributes, part and pose. In *ICCV*, 2015. 1, 2

[25] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014. 5

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 1988. 2

[27] C. Shi, C. Wang, B. Xiao, Y. Zhang, S. Gao, and Z. Zhang. Scene text recognition using part-based tree-structured character detection. In *CVPR*, 2013. 7, 8

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 8

[29] X. Song, T. Wu, Y. Jia, and S.-C. Zhu. Discriminatively trained and-or tree models for object detection. In *CVPR*, 2013. 1, 4, 8

[30] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014. 6

[31] B. Su, S. Lu, S. Tian, J. H. Lim, and C. L. Tan. Character recognition in natural scenes using convolutional co-occurrence hog. In *ICPR*, 2014. 7, 8

[32] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*, 2014. 2

[33] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 2013. 8

[34] L. Wan, D. Eigen, and R. Fergus. End-to-end integration of a convolution network, deformable parts model and non-maximum suppression. In *CVPR*, 2015. 2, 8

[35] J. Wang and A. L. Yuille. Semantic part segmentation using compositional model combining shape and appearance. In *CVPR*, 2015. 1, 2, 4

[36] C. K. Williams and M. K. Titsias. Greedy learning of multiple objects in images using robust statistics and factorial learning. *Neural Computation*, 2004. 2

[37] T. Wu, B. Li, and S. C. Zhu. Learning and-or model to represent context and occlusion for car detection and viewpoint estimation. *PAMI*, 2016. 1, 2, 4

[38] T. Wu, Y. Lu, and S. C. Zhu. Online object tracking, learning and parsing with and-or graphs. *PAMI*, 2016. 2, 4

[39] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 2, 8

[40] Q. Zhang, Y. Nian Wu, and S.-C. Zhu. Mining and-or graphs for graph matching and object discovery. In *ICCV*, 2015. 2

[41] L. Zhu, Y. Chen, A. Yuille, and W. Freeman. Latent hierarchical structural learning for object detection. In *CVPR*, 2010. 1, 2, 4, 8

[42] L. L. Zhu, Y. Chen, A. Torralba, W. Freeman, and A. Yuille. Part and appearance sharing: Recursive compositional models for multi-view multi-object detection. In *CVPR*, 2010. 1, 2, 5

[43] L. L. Zhu, Y. Chen, and A. Yuille. Recursive compositional models for vision: Description and review of recent work. *Journal of Mathematical Imaging and Vision*, 2011. 1, 2, 4, 5

[44] S.-C. Zhu and D. Mumford. *A stochastic grammar of images*. Now Publishers Inc, 2007. 1, 2, 4