Learning a Hierarchical Compositional Shape Vocabulary for Multi-class Object Representation

Sanja Fidler, Student Member, IEEE, Marko Boben, and Aleš Leonardis, Member, IEEE

Abstract—Hierarchies allow feature sharing between objects at multiple levels of representation, can code exponential variability in a very compact way and enable fast inference. This makes them potentially suitable for learning and recognizing a higher number of object classes. However, the success of the hierarchical approaches so far has been hindered by the use of hand-crafted features or predetermined grouping rules. This paper presents a novel framework for learning a hierarchical compositional shape vocabulary for representing multiple object classes. The approach takes simple contour fragments and learns their frequent spatial configurations. These are recursively combined into increasingly more complex and class-specific shape compositions, each exerting a high degree of shape variability. At the top-level of the vocabulary, the compositions are sufficiently large and complex to represent the whole shapes of the objects. We learn the vocabulary layer after layer, by gradually increasing the size of the window of analysis and reducing the spatial resolution at which the shape configurations are learned. The lower layers are learned jointly on images of all classes, whereas the higher layers of the vocabulary are learned incrementally, by presenting the algorithm with one object class after another. The experimental results show that the learned multi-class object representation scales favorably with the number of object classes and achieves a state-of-the-art detection performance at both, faster inference as well as shorter training times.

Index Terms—Hierarchical representations, compositional hierarchies, unsupervised hierarchical structure learning, multiple object class recognition and detection, modeling object structure.

1 INTRODUCTION

VISUAL categorization of objects has been an area of active research in the vision community for decades. Ultimately, the goal is to recognize and detect an increasing number of object classes in images within an acceptable time frame. The problem entangles three highly interconnected issues: the internal object representation which should compactly capture the high visual variability of objects and generalize well over each class, means of learning the representation from a set of images with as little supervision as possible, and an effective inference algorithm that robustly matches the object representation against the image.

Using vocabularies of visual features has been a popular choice of object class representation and has yielded some of the most successful performances for object detection to date [1], [2], [3], [4], [5], [6]. However, the majority of these works are currently using flat coding schemes where each object is represented with either no structure at all by using a bag-of-words model or only simple geometry induced over a set of intermediately complex object parts. In this paper, our aim is to model the hierarchical compositional structure of the objects and do so for multiple object classes.

Modeling structure (geometry of objects) is important for several reasons. Firstly, since objects within a class have usually distinctive and similar shape, it allows for an efficient shape parametrization with good generalization capabilities. It further enables us to parse objects into meaningful components which is of particular importance in robotic applications where the task is not only to detect objects but also to execute higher-level cognitive tasks (manipulations, grasping, etc). Thirdly, by inducing the structure over the features the representation becomes more robust to background clutter.

Hierarchies incorporate structural dependencies among the features at multiple levels: objects are defined in terms of parts, which are further composed from a set of simpler constituents, etc [7], [8], [9], [10], [11]. Such architectures allow sharing of features between the visually similar as well as dissimilar classes at multiple levels of specificity [12], [13], [11], [10]. Sharing of features means sharing common computations and increasing the speed of the joint detector [2]. More importantly, shared features lead to better generalization [2] and can play an important role of regularization in learning of novel classes with few training examples. Furthermore, since each feature in the hierarchy recursively models certain variance over its parts, it captures a high structural variability and consequently a smaller number of features are needed to represent each class.

Learning of feature vocabularies without or with little supervision is of primary importance in multi-class object representation. By learning, we minimize the amount of time of human involvement in object labeling [9], [14] and avoid bias of pre-determined grouping rules or manually crafted features [15], [16], [17], [18]. Second, learning the representation statistically yields features

---

*S. Fidler, M. Boben and A. Leonardis are with the Faculty of Computer and Information Science, University of Ljubljana, Slovenia. E-mail: {sanja.fidler,marko.boben,ales.leonardis}@fri.uni-lj.si
most shareable between the classes, which may not be well predicted by human labelers [19]. However, the complexity of learning the structure of a hierarchical representation bottom-up and without supervision is enormous: there is a huge number of possible feature combinations, the number of which exponentially increases with each additional layer — thus an effective learning algorithm must be employed.

In this paper, the idea is to represent the objects with a learned hierarchical compositional shape vocabulary that has the following architecture. The vocabulary at each layer contains a set of hierarchical deformable models which we will call compositions. Each composition is defined recursively: it is a hierarchical generative probabilistic model that represents a geometric configuration of a small number of parts which are themselves hierarchical deformable models, i.e., compositions from a previous layer of the vocabulary. We present a framework for learning such a representation for multiple object classes. Learning is statistical and is performed bottom-up. The approach takes simple oriented contour fragments and learns their frequent spatial configurations. These are recursively combined into increasingly more complex and class-specific shape compositions, each exerting a high degree of shape variability. In the top-level of the vocabulary, the compositions are sufficiently large and complex to represent the whole shapes of the objects. We learn the vocabulary layer after layer, by gradually increasing the size of the window of analysis and the spatial resolution at which the shape configurations are learned. The lower layers are learned jointly on images of all classes, whereas the higher layers of the vocabulary are learned incrementally, by presenting the algorithm with one object class after another. We assume supervision in terms of a positive and a validation set of class images — however, the structure of the vocabulary is learned in an unsupervised manner. That is, the number of compositions at each layer, the number of parts for each of the compositions along with the distribution parameters are inferred from the data without supervision.

We experimentally demonstrate several important issues: 1.) Applied to a collection of natural images, the approach learns hierarchical generative models for various curvatures, T- and L-junctions, i.e., features usually emphasized by the Gestalt theory [20]; 2.) We show that these generic compositions can be effectively used for object classification; 3.) For object detection we demonstrate a competitive speed of detection with respect to the related approaches already for a single class. 4.) For multi-class object detection we achieve a highly sub-linear growth in the size of the hierarchical vocabulary at multiple layers and, consequently, a scalable complexity of inference as the number of modeled classes increases; 5.) We demonstrate a competitive detection accuracy with respect to the current state-of-the-art. Furthermore, the learned representation is very compact — a hierarchy modeling 15 object classes uses only 1.6Mb on disk.

The remainder of this paper is organized as follows. In Section 2 we review the related work. Section 3 presents our hierarchical compositional representation of object shape with recognition and detection described in Section 4. In Section 5 our learning framework is proposed. The experimental results are presented in Section 6. The paper concludes with a summary and discussion in Sec. 7 and pointers to future work in Sec. 8.

2 RELATED WORK AND CONTRIBUTIONS

Compositional hierarchies. Several compositional approaches to modeling objects have been proposed in the literature, however, most of them relied on hand-crafted representations, pre-determined grouping rules or supervised training with manually annotated object parts [9], [16], [17], [21]. The reader is referred to [9], [22] for a thorough review.

Unsupervised learning of hierarchical compositional vocabularies. Work on unsupervised hierarchical learning has been relatively scarce. Utans [23] has been the first to address unsupervised learning of compositional representations. The approach learned hierarchical mixture models of feature combinations, and was utilized on learning simple dot patterns.

Based on the Fukushima’s model [24], Riesenhuber and Poggio [25] introduced the HMAX approach which represents objects with a 2-layer hierarchy of Gabor feature combinations. The original HMAX used a vocabulary of pre-determined features, while these have subsequently been replaced with randomly chosen templates [26]. Since no statistical learning is involved, as much as several thousands of features are needed to represent the objects. An improved learning algorithm has recently been proposed by Masquelier and Thorpe [27].

Among the neural network representatives, Convolutional nets [28], [29] have been most widely and successfully applied to generic object recognition. The approach builds a hierarchical feature extraction and classification system with fast feed-forward processing. The hierarchy stacks one or several feature extraction stages, each of which consists of filter bank layer, non-linear transformation layers, and a pooling layer that combines filter responses over local neighborhoods using an average or max operation, thereby achieving invariance to small distortions [29]. A similar approach is proposed by Hinton [30] with recent improvements by Ng et al. [31]. One of the main drawbacks of these approaches, however, may be that they do not explicitly model the spatial relations among the features and are thus less robust to shape variations.

Bouchard and Triggs [7] proposed a 3-layer hierarchy (extension of the constellation model [1]) and a similar representation was proposed by Torralba et al. [2], [32]. For tractability, all of these models are forced to use very sparse image information, where prior to learning and detection, a small number (around 30) of interest points are detected. Using highly discriminative SIFT features might limit their success in cluttered images or
on structurally simpler objects with little texture. The repeatability of the SIFT features across the classes is also questionable [26]. On the other hand, our approach is capable of dealing with several tens of thousands of contour fragments as input. We believe that the use of repeatable, dense and indistinctive contour fragments provide us with a higher repeatability of the subsequent object representation facilitating a better performance.

Epshtein and Ullman [33] approached the representation from the opposite end; the hierarchy is built by decomposing object relevant image patches into recursively smaller entities. The approach has been utilized on learning each class individually while a joint multi-class representation has not been pursued. A similar line of work was adopted by Mikolajczyk et al. [34] and applied to recognize and detect multiple (5) object classes.

Todorovic and Ahuja [35] proposed a data-driven approach where a hierarchy for each object example is generated automatically by a segmentation algorithm. The largest repeatable subgraphs are learned to represent the objects. Since bottom-up processes are usually unstable, exhaustive grouping is employed which results in long training and inference times.

Ommer and Buhmann [11] proposed an unsupervised hierarchical learning approach, which has been successfully utilized for object classification. The features at each layer are defined as histograms over a larger, spatially constrained area. Our approach explicitly models the spatial relations among the features, which should allow for a more reliable detection of objects with lower sensitivity to background clutter.

The learning frameworks most related to ours include the work by [36], [37] and just recently [38]. However, all of these methods build separate hierarchies for each object class. This, on the one hand, avoids the massive number of possible feature combinations present in diverse objects, but, on the downside, does not exploit the shareability of features among the classes.

Our approach is also related to the work on multi-class shape recognition by Amit et al. [39], [40], [13], and some of their ideas have also inspired our approach. While the conceptual representation is similar to ours, the compositions there are designed by hand, the hierarchy has only three layers (edges, parts and objects), and the application is mostly targeted to reading licence plates.

While surely equally important, the work on learning of visual taxonomies [41], [42] of object classes tackles the categorization/recognition process by hierarchical cascade of classifiers. Our hierarchy is compositional and generative with respect to object structure and does not address the taxonomic organization of object classes.

Contour-based recognition approaches. Since our approach deals with object shape, we also briefly review the related work on contour-based object class detection.

Contour fragments have been employed in the earlier work by Selinger and Nelson [43] and a number of follow-up works have used a similar approach. Opelt et al. [3] learned boundary fragments in a boosting frame-
The geometric configuration of parts is modeled by only one of the compositions in the vocabulary. The number of top-layer compositions will be used to represent the whole distribution of shapes in the class, i.e., the final, object layer the compositions will represent the shapes of the objects. The set and structure of all compositions in the vocabulary is further discussed in Section 5.3.

We will use the following notation. Let \( \Omega \) denote the set and structure of all compositions in the vocabulary and \( \Theta \) their parameters. Since the vocabulary has a hierarchical structure we will write it as \( \Omega = \Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega^n \cup \Omega^C \) where \( \Omega^i = \{ \omega^i \} \) is a set of compositions at layer \( \ell \). Whenever it is clear from the context, we will omit the index \( i \). With \( \Omega^C \) we denote the top, object layer of compositions (we will use \( O = 6 \) in this paper, and this choice is discussed in Section 5.3), which roughly code the whole shapes of the objects. The final, object class layer \( \Omega^C \) is not compositional, but only pools all of the corresponding object layer compositions for each class separately. Specifically, a model for a particular object class \( c \in \Omega^C \) is represented as a disjunction (OR) of a (sub)set of object layer compositions.

The definition of a composition with respect to just one layer below is akin to that of the constellation model [45]. A composition \( \omega^\ell \), where \( \ell > 1 \), consists of \( P \) parts (\( P \) can be different for different compositions) with appearances \( \theta_{\text{app}}^\ell = [\theta_{\text{app}}^j]^P_{j=1} \) and geometric parameters \( \theta_g^\ell = [\theta_{\text{app}}^j]^P_{j=1} \). The appearance \( \theta_{\text{app}}^j \) of a part is a discrete distribution over the compositions from the previous layer. For example, \( \theta_{\text{app}}(\omega_{k-1}^j) = 0.8 \) means that the \( j \)-th part of \( \omega^\ell \) is 0.8 likely to be the \( k \)-th composition \( \omega_{k-1}^j \) from layer \( \ell - 1 \). The geometric relations between the position of each of the parts relative to the reference part are modeled by two-dimensional Gaussians with parameters \( \theta_g^\ell = (\mu_g^\ell, \Sigma_g^\ell) \). For the convenience of notation later on, we will also represent the location of a reference part with respect to itself with a Gaussian having zero mean and small variance, \( c^2\text{Id} \).

The models at the first layer of the vocabulary are defined over the space of image features, which will in this paper be \( n \)-dimensional Gabor feature vectors (explained in Sec. 4.1). Each model \( \omega^1 \) has an appearance distribution over the feature vectors, which is taken to be an \( n \)-dimensional Gaussian with parameters \( (\mu^1, \Sigma^1) \).

Lastly, the class layer models are only specified by an appearance distribution over the object layer compositions. A class model \( c \) thus has the appearance distribution \( \theta_{\text{app}}^c \), which is non-zero only for those object layer compositions that represent the object shapes in class \( c \). We will denote this set by \( \{ \omega^i \} \), \( i_c \subset \Omega^C \), with \( \theta_{\text{app}}^c(\omega^i) > 0 \) for each \( i_c \) and \( \theta_{\text{app}}^c(\omega^i) = 0 \) for all other \( i \). The sets of object layer compositions for two
different classes are always disjunct.

Note that each composition can only generate shapes within a limited spatial extent, that is, the window around a generated shape is of limited size and the size is the same for all compositions at a particular layer $\ell$ of the vocabulary. We will denote it with $\gamma^\ell$. The limiting size $\gamma^\ell$ increases exponentially with the level of hierarchy. Figure 2 depicts the gradual increase in complexity and size of the compositions in the hierarchy.

### 4 Recognition and detection

The model is best explained by first considering inference. Thus, let us for now assume that the representation is already known and we are only concerned with recognizing and detecting objects in a query image given the model. How we learn the representation will be explained in Sec. 5.

Subsec. 4.1 explains the image features we use, which form a set of observations, and how we extract them from an image. Subsec. 4.2 describes the inference of the hidden states of the model, while Subsec. 4.2.3 shows how to perform it in a computationally efficient way. In Subsec. 4.3, we explain how we detect objects given the inferred hidden states.

#### 4.1 Extracting image features

Let $I$ denote a query image. The features which we extract from $I$ should depend on how we define the base models at the first layer $\Omega^1$ of the vocabulary. In this paper we choose to use oriented edge fragments, however, the learning and the recognition procedures are general and independent of this particular choice.

In order to detect oriented edges in an image we use a Gabor filter bank:

$$g_{\lambda,\varphi,\gamma}(x, y, \psi) = e^{-\frac{x^2+\gamma^2 y^2}{2\sigma^2}} \cos \left( \frac{2\pi u}{\lambda} + \varphi \right)$$

$$u = x \cos \psi - y \sin \psi, \quad v = x \sin \psi + y \cos \psi,$$

where $(x, y)$ represents the center of the filter’s domain, and the parameters in this paper are set to $(\lambda, \gamma, \sigma) = (6, 0.75, 2)$. A set of two filter banks is used, one with even, $\varphi = 0$, and the other with odd, $\varphi = -\frac{\pi}{2}$, Gabor kernels defined for $n$ equidistant orientations, $\psi = \frac{i\pi}{n}$, $i = 0, 1, \ldots, n - 1$. For the experiments in this paper we use $n = 6$.

We convolve the image $I$ with both filter banks and compute the total energy for each of the orientations [49]:

$$\mathcal{E}(x, y, \psi) = \sqrt{r_0^2(x, y, \psi) + r_{-\pi/2}^2(x, y, \psi)}, \quad (1)$$

where $r_0(x, y, \psi)$ and $r_{-\pi/2}(x, y, \psi)$ are the convolution outputs of even and odd Gabor filters at location $(x, y)$ and orientation $\psi$, respectively. We normalize $\mathcal{E}$ to have the highest value in the image equal to $1$. We further perform a non-maxima suppression over the total energy $\mathcal{E}$ to find the locations of the local maxima for each of the orientations. This is similar to performing the Canny operator and taking the locations of the binary responses. At each of these locations $(x, y)$, the set of which will be denoted with $X = \{(x, y)\}$, we extract Gabor features $f = f(x, y)$ which are $n$-dimensional vectors containing the orientation energy values in $(x, y)$, $f(x, y) = [\mathcal{E}(x, y, i\pi/n)]_{i=0}^{n-1}$. Specifically, the feature set $F$ is the set of all feature vectors extracted in the image, $F = \{f(x, y), (x, y) \in X\}$. The number of features in $F$ is usually around $10^4$ for an image of average size and texture. The image features $(F, X)$ serve as the observations to the recognition procedure. Figure 3 shows an example of feature extraction.

**Scale.** The feature extraction as well as the recognition process is performed at several scales of an image: we use a Gaussian pyramid with two scales per octave and process each scale separately. That is, feature extraction and recognition up to the object level $O$ are performed at every scale and independently of other scales. For detection of object classes we then consider information from all scales. Not to overload the notation, we will just refer to one scale of $I$.

#### 4.2 Inference

Performing inference in a query image with a given vocabulary entails inferring a hierarchy of hidden states from the observations (which are the image features $(F, X)$). The hidden states at the first layer will be the only ones receiving input directly from the observations,
whereas all the other states will receive input from a hidden layer below.

Each hidden state $z^f$ has two variables $z^f = (\omega^f, x^f)$, where $\omega^f$ denotes a composition from the vocabulary and $x^f$ a location in an image (to abbreviate notation from here on, we will use $x$ instead of $(x, y)$ to denote the location vector). Note that $\omega^f$ is in fact a hierarchical probabilistic model, so the first variable is just a reference to the model. The notation $x^f$ is used to emphasize that $x$ is from a spatial grid with resolution corresponding to hidden layer $\ell$, i.e., the spatial resolution of the locations of the hidden states will be increasingly coarser with the level of hierarchy (discussed in Sec. 4.2.3).

We address three problems:

1) **Image likelihood.** Calculating the likelihood of an observed image neighborhood $J$ under a particular composition (hierarchical model), say $\omega^f$: $p(J|\omega^f)$.

2) **Most probable hidden state activation.** Finding the most likely tree $T^*(J, \omega^f)$ of hidden state activations to have generated an observed image neighborhood $J$ given the model $\omega^f$: $T^*(J, \omega^f) = \arg \max_T p(T|J; \omega^f)$.

3) **Multi-class object detection and recognition.** Finding the interpretation of an image in terms of a collection of objects in terms of:

   a) Finding the neighborhoods $J$ of objects and the class $c$ they belong to: $\{(J_i, c_i) : p(J_i|c_i) > \tau_c \text{ and } J_i \cap J_k = \emptyset; \forall (i, k), i \neq k\}$, where $\tau_c$ is a class-specific threshold that separates foreground (class $c$) against the background.

   b) Finding object segmentations which will be defined by the leaves (the observations) of the most probable hidden state tree activations $T^*(J_i, c_i)$ for the detected objects $(J_i, c_i)$.

### 4.2.1 Calculating the likelihood

Since we will deal with images of much larger size than $r^f$, it is convenient to define $I(x^f)$ to be a neighborhood in an image $I$, centered in $x^f$ and radius $r^f$:

$$I(x^f) = \{ (f, x) \in (F, X) : ||x - x^f||_2 \leq r^f \}$$  \hspace{1cm} (2)

A neighborhood $I(x^f)$ is thus defined as a collection of features in a circular region around $x^f$.

For a given composition, e.g. $\omega^f$, we would like to calculate the likelihood $p(I(x^f)|\omega^f)$ of an observed image neighborhood $I(x^f)$ under $\omega^f$. By denoting $z^f = (\omega^f, x^f)$ as a hidden state centered in $x^f$ (in the center of the neighborhood), then the following formulations are equivalent and will be used interchangeably throughout the paper: $p(I(x^f)|\omega^f) = p(I(x^f), z^f|\omega^f) = p(I(x^f)|z^f)$.

Next, let $T(z^f)$ denote a tree of hidden states where the root (top) node is $z^f$ and the leaves (layer 0) are the observations. To calculate the likelihood, we thus need to sum over all possible such trees:

$$p(I(x^f)|\omega^f) = \sum_{T(z^f)} p(I(x^f), T(z^f)|\omega^f) \tag{3}$$

Since there is an exponential number of possible trees of hidden states, calculating this expression directly is intractable. However, we can exploit the recursive definition of the compositions to get also a recursive formulation for the likelihood.

We first write the joint distribution. Let $T^\ell$ denote the set of states at layer $\ell'$ of the tree $T(z^f)$, let $T(z^f)$ be the subtree of $T(z^f)$ that has the root $z^f \in T^\ell$, and let $T^\ell_1 = [z^f_{\ell_1} := (\omega^f, x^f)]_{\ell_1=1}^{\ell_1}$ be the hidden states at layer $\ell_1-1$ of the tree (just one layer below the root). We will re-write the joint $p(I(x^f), T(z^f)|\omega^f)$ into a recursive formulation. We first factorize it in the following way

$$p(I(x^f), T(z^f)|\omega^f) = \sum_{j=1}^{P} p_{F}(I(x^f), T(z^f_j^\ell), z^f_{\ell_1}|z^f_1) \prod_{k=P+1}^{T_1} p_{B}(I(x^f), T(z^f_k), z^f_{k-1}|z^f_k) \tag{4}$$

The foreground term can be further factorized:

$$p_{F}(I(x^f), T(z^f_j^\ell), z^f_{\ell_1}|z^f_1) = p_{F}(I(x^f), T(z^f_{\ell_1}), z^f_{\ell_1}|z^f_1; \omega^f)$$  \hspace{1cm} (6)

Since conditioning on layer $\ell - 1$ renders lower layers independent of $\ell$, the second term becomes

$$p_{B}(I(x^f), T(z^f_{\ell_1}), z^f_{\ell_1}|z^f_{j}) = p(I(x^f), T(z^f_{\ell_1})|z^f_{j}; \omega^f)$$  \hspace{1cm} (7)

Given how we defined the compositions, the term $p_{F}(z^f_{\ell_1}|z^f_1)$ factorizes into appearance and geometry:

$$p_{F}(z^f_{\ell_1}|z^f_1) = p(\omega^f_{\ell_1} | \theta^f_{\omega, j}) p(x^f_{\ell_1} | x^f_1; \theta^f_{x,y})$$

$$= \theta^f_{\omega, j}(\omega^f_{\ell_1}) \mathcal{N}(x^f_{\ell_1} - x^f_1 | \mu^f_{\omega, j}, \Sigma^f_{\omega, j}) \tag{8}$$

where the first term is the probability of $j$-th part under $\omega^f$ having the appearance $\omega^f_{\ell_1}$ (see Sec. 3). The second term is the spatial compatibility between the location of state $z^f_{\ell_1}$ and location $x^f$ of the parent $z^f_1$ under the model for part $j$.

Similarly as (6), the background term in (4) factorizes:

$$p_{B}(I(x^f), T(z^f_{\ell_1}), z^f_{\ell_1}|z^f_{j})$$

$$= p_{B}(z^f_{\ell_1}|z^f_{j}) p_{B}(I(x^f), T(z^f_{\ell_1})|z^f_{\ell_1}; \omega^f)$$

$$= \alpha \cdot (1 - p(I(x^f_{\ell_1}), T(z^f_{\ell_1})|\omega^f_{\ell_1})) \tag{9}$$

where the probability $p_{B}(z^f_{\ell_1}|z^f_{j})$ that a state $z^f_{\ell_1}$ belongs to the background is taken to be $\alpha$, and the likelihood of the observations under a background distribution is taken to be $1- \text{the likelihood of the same observation to be the foreground}$.
The joint distribution in (4) can now be rewritten in a recursive form:

\[
p(I(x^\ell), T|\omega^\ell) = \left( \prod_{j=1}^{P} \theta_{app}^\ell(\omega_{j}^{\ell-1}) N(x_j^{\ell-1} - x^\ell | \mu_j^\ell, \Sigma_j^\ell) \cdot p(I(x_j^{\ell-1}), T(z_j^{\ell-1})|\omega_j^{\ell-1}) \right) \\
\cdot \left( \alpha^N_B \prod_{k=P+1}^{|T^{\ell-1}|} (1 - p(I(x_k^{\ell-1}), T(z_k^{\ell-1})|\omega_k^{\ell-1})) \right)
\]

where \( N_B = |T^{\ell-1}| - P \) is the number of hidden states in \( T^{\ell-1} \) belonging to the background.

Due to the recursive formulation of the joint, we can now also get a recursive formulation for the likelihood in (3). Let us define \( q(I,z) = p(I(x)|\omega) \), where \( z = (\omega, x) \). By manipulating the sums in (3) and (10), we obtain the following form for the likelihood:

\[
q(I,z) = \sum_{T^{\ell-1}} \left[ \left( \prod_{j=1}^{P} \theta_{app}^\ell(\omega_{j}^{\ell-1}) N(x_j^{\ell-1} - x^\ell | \mu_j^\ell, \Sigma_j^\ell) \cdot q(I, z_j^{\ell-1}) \right) \cdot \alpha^N_B \prod_{k=P+1}^{|T^{\ell-1}|} (1 - q(I, z_k^{\ell-1})) \right] \quad \text{(11)}
\]

To complete the recursion we only need to define the data term:

\[
q(I,z^\ell) = p(I(x^\ell) | z^\ell) = p(f(x^\ell) | z^\ell) = N(f(x^\ell) | \mu^1, \Sigma^1) \quad \text{(12)}
\]

\( q(I,z) \) is a likelihood of a feature vector \( f \) (which is in our case a 6-dimensional vector of normalized Gabor filter outputs) to be generated with a model \( \omega^1 \) for a particular edge orientation. It is modeled with a (6-dimensional) Gaussian distribution.

The likelihoods can thus be computed efficiently by dynamic programming, similarly as in the forward algorithm for Hierarchical HMM [50]. This procedure along with further speed-ups inspired by the coarse-to-fine search of Amit and Geman et al. [39], [37], [51] is described in Subsec. 4.2.3.

### 4.2.2 Finding most probable hidden state activation

Given a model \( \omega^\ell \) we would additionally like to find the most probable tree of hidden states that generated an image neighborhood \( I(x^\ell) \):

\[
T^\star(z^\ell) = \arg \max_{T(z^\ell)} q(T(z^\ell) | I(x^\ell); \omega^\ell) = \arg \max_{T(z^\ell)} p(I(x^\ell), T(z^\ell) | \omega^\ell) \quad \text{(13)}
\]

If we define \( \delta(I,z) = \max_{T(z)} p(I(x), T(z)|\omega) \), we can write a recursion for \( \delta(I,z^\ell) \) similar to that in (11), however, with max instead of the sum. For the data term we have: \( \delta(I,z^\ell) = p(f(x^\ell) | z^\ell) \). The tree \( T^\star(z^\ell) \) can then be found by dynamic programming, similarly as in the Viterbi recurrence for the Hierarchical HMM [50]; we compute \( \delta(I,z^\ell) \) sequentially, from bottom layer to top and for each hidden state we keep back-pointers to the states at the previous layer that produced the maximum. The complete tree \( T^\star(z^\ell) \) can then be found by tracing back the back-pointers in the resulting graph.

We will refer to the leaves of the tree \( T^\star(z^\ell) \) as the support of a hidden state \( z^\ell \), and denote it with \( \text{supp}(z^\ell) \).

### 4.2.3 Efficient computation

This section explains how to calculate the likelihoods and the most probable trees efficiently. We will employ the following approach:

- **Recursive bottom-up computation**: Compute the likelihoods of the hidden states in a recursive bottom-up fashion, where the intermediate computations are shared among higher layer hidden states.
- **Coarse-to-fine computation**: Prior to computing the likelihood of a state we will first perform a learned computationally inexpensive test upon which we will decide whether the likelihood of that state will be computed or whether the state will be pruned.

Since ultimately, we want to recognize and detect objects in images, our goal is to compute the image likelihoods under each of the object-layer compositions \( \{\omega^O_i\} \) and do so for all image neighborhoods \( \{I(x^O_i)\} \) (we will show how the classes of the objects can be inferred based on these likelihoods in Sec. 4.3). In order to compute the likelihood for a particular state \( z^O_{ij} = (\omega^O_i, x^O_j) \), the recursive formulation in (11) implies a bottom-up computation: start by computing the likelihoods of the hidden states at the first layer and continuing up. Note that we do not need to compute the likelihoods of the intermediate hidden states for each \( z^O_{ij} \) separately, but can compute each of them only once (different states defined over overlapping neighborhoods can re-use the intermediate hidden states).

Notice, however, that this entails computing the likelihoods \( q(I,z^\ell) \) for all possible locations \( x^\ell \) over the image and all compositions \( \omega^\ell \) from the vocabulary, and for each of the layers \( \ell = 1, \ldots, \ell \), which would still cause inference to run slow. To make the computations efficient, we will exploit the fact that most hidden states will be highly unlikely. We will make use of an idea of coarse-to-fine search proposed by Amit and Geman et al. [39], [37], [51]. The strategy will be to prune the unpromising hidden states based on computationally inexpensive tests. A test is a simple decision function employed prior to computing the likelihood of a state, which tells us whether the state is worth considering at all. In the case the test will return a positive answer, the likelihood for a particular state will be computed, while otherwise the state will be pruned. Note that due to pruning, the resulting likelihoods in the hierarchy will not be exact. However, since only the very unlikely states will be pruned, the computed likelihoods will represent a sufficiently good approximation. We explain the complete procedure next.

Let \( G = (I) = (Z,E) \) denote an inference graph, where the nodes \( Z \) of the graph are the ("promising") hidden
states $z^t$. With each state $z^t$ we also associate two values: the
likelihood $q(I, z^t)$ and the probability of the most
probably hidden tree $\delta(I, z^t)$. Graph $\mathcal{G}$ has a hierarchical
structure where the vertices $Z$ are partitioned into vertex
layers $1$ to $O$, that is, $Z = Z^0 \cup Z^1 \cup \ldots \cup Z^O$. The vertex
layer $Z^\ell$ contains the hidden states $z^t$ from layer $\ell$, while
$Z^0$ contains the observations $(F, X)$ (each $(f, x) \in (F, X)$
is a node in $Z^0$).

Graph $\mathcal{G}$ is built from bottom (layer $1$) to top (layer $O$).
For a hidden state $z^t = (\omega^t, x^t)$ we first perform a simple
learned test $T_{\omega^t}(I, z^t)$ that has the following form:
\begin{equation}
T_{\omega^t}(I, z^t) = 1 (\delta(I, z^t) > \tau_{\omega^t}),
\end{equation}
where $1$ is an indicator function and $\tau_{\omega^t}$ is a learned
threshold. A notation $\tau_{\omega^t}$ is used, indicating that each
composition $\omega^t$ in the vocabulary can have its own
threshold. If $T$ returns a positive answer then we add the
hidden state $z^t$ to $Z^\ell$ and calculate its likelihood
$q(I, z^t)$. In the case when $T = 0$, the state $z^t$ is not
considered further, i.e., not added to the inference graph.

Intuitively, a hidden state is pruned if only if even the
most probable hidden tree for that state has a very
small probability. During the process of training (until
the hierarchy is built up to the level of objects), these
thresholds are fixed and set to very small values (in this
paper we use $\tau = 0.05$). Their only role is to prune the
very unlikely hypotheses and thus both, speed-up
computation as well as minimize the memory/storage
requirements (the size of $G$). After the class models (the
final, layer $O$ of the vocabulary) are learned we can also
learn the thresholds in a way to optimize the speed of
inference while retaining the detection accuracy, as will
be explained in Sec. 5.3.1.

For each state added to $Z^\ell$ we additionally form edges in
$E$ from the state node $z^t$ to the states $T_{\omega^t}^{\ell-1}(z^{t-1}) =
[z^{t-1}_j] \in Z^{\ell-1}$ that produced the value $\delta(I, z^t)$.

Reduction in spatial resolution. After each layer $Z^\ell$ is
built, we perform spatial contraction, where the locations
$x^t$ of the hidden states $z^t$ are downsampled by a factor
$\rho^t < 1$. Among the states that code the same composition
$\omega^t$ and for which the locations (taken to have integer
values) become equal, only the state with the highest
likelihood is kept. We use $\rho^1 = 1$ and $\rho^\ell = 0.5$, $\ell > 1$.
This step is mainly meant to reduce the computational
load: by reducing the spatial resolution at each layer, we
bring the far-away (location-wise) hidden states closer.
This, indirectly (through learning), keeps the scales of the
Gaussians relatively small across all compositions in
the vocabulary and makes inference faster.

4.3 Object class detection and recognition

This section explains how to perform
1) multi-class object recognition and detection,
2) segmentation and parsing of the detected objects
once we have obtained (by using the procedures de-
scribed in the previous sections) the inference graph
$\mathcal{G}$ built to hidden layer $Z^O$.

We first show how to compute the state likelihoods
under the class models from the vocabulary (layer $O$)
given $G$. Based on these likelihoods we make a decision
on the presence of an object at a certain location. An
object detection will, in this paper, taken to be a hidden
state $z^C = (c, x^C)$ with likelihood $q(I, z^C)$ higher than
a class-specific threshold, that is, a threshold is used to
decide whether a certain state belongs to the foreground
(object) or background. Once this simple decision is
made we could additionally use a more sophisticated
verification step, e.g., employing SVM over the features
in the detected region (as in [52], for example), however,
we have not employed one in this paper.

Let $Z^C$ be a “class” hidden layer of $G$ and let $z^C =
(c, x^C) \in Z^C$ be a hidden state that represents the
presence of an object class $c$ at location $x^C$. Following (11),
we can compute its likelihood $q(I, z^C)$ as follows:
\begin{equation}
q(I, z^C) = p(I(x^C) \mid z^C) = \sum_{z_i^C = (c_i^C, x_i^C)} p(z_i^C \mid z^C) q(I, z_i^0) = \sum_{z_i^C} \theta_{\text{app}}(\omega_i^C) q(I, z_i^C)
\end{equation}
Note that the sum runs over all possible states $z_i^0$ at
location $x^C$. However, since $\theta_{\text{app}}$ is zero for all but a
subset of layer $O$ compositions, i.e., $\{\omega_i^C\}$, we only need
to sum over the corresponding hidden states $z_i^0$.

For object detection, we will only keep those states $z^C
that have likelihoods higher than a threshold, $q(I, z^C) >
\tau_c$. Note that $\tau_c$ is a class-specific threshold, meaning that
different decisions are made for different classes. The
hidden states not satisfying the decision inequality are
removed from $Z^C$.

However, all nodes from $Z^C$ do not yet represent the
final object detections, but define a set of object hypotheses.
Since different neighborhoods in an image were assumed
independent during inference, we can potentially have
multiple different hypotheses at different locations in $Z^C$
explaining partly the same image area, and thus a
selection between the hypotheses is necessary. This is
illustrated in Fig. 4: the left side shows the supports of
object hypotheses for the horse class and the right side
for the giraffe class. Ultimately, we would like to select
the giraffe detection over the horse one, since it explains
a larger portion of the image, even though its likelihood
might be somewhat lower than that under the horse
hypothesis. For this reason, we will perform competition
among overlapping hypotheses as explained below.

Competition between overlapping hypotheses. Let $z_i^C$ and
$z_j^C$ denote two object hypotheses with overlapping
supports. To choose between them, we evaluate both
hypotheses on the union of their supports, i.e., $\text{supp}(z_i^C) \cup
\text{supp}(z_j^C)$. The unexplained features under each of the
hypotheses count as background. We define the cost of a
hypothesis with respect to the other as
\begin{equation}
C(z_i^C \mid z_j^C) = |\text{supp}(z_i^C) \cdot \log q(I, z_i^C) +
|\text{supp}(z_j^C) \setminus \text{supp}(z_i^C)| \cdot \log \alpha.
\end{equation}
The cost weights each feature in the support of a hypothesis \( z^C \) by the log-likelihood \( q(I, z^C) \) and the background features (the features in \( \text{supp}(z^C) \cup \text{supp}(z^O) \)) not explained by \( z^C \) by \( \log \alpha \). The hypothesis with the lower cost is selected as an object detection. The selection process is run in a greedy fashion until all overlapping hypotheses are accounted for.

A parse tree (most probable tree of hidden states) for each of the object detections \( z^C \) is obtained by tracing back the edges in \( \mathcal{G} \), starting from the root node \( z^C \) down. A few examples of parse trees are shown in Fig. 15 in Sec. 6. The segmentation of the object is defined by the leaves of the tree, i.e., the support of \( z^C \).

5 Learning

This section addresses the problem of learning the representation, which entails the following:

1) **Learning the vocabulary of compositions.** Finding an appropriate set of compositions to represent our data and learning the structure of each of the compositions (the number of parts and a (rough) initialization of the parameters).

2) **Learning the parameters of the representation.** Finding the parameters of geometry and appearance for each composition as well as the thresholds to be used in the approximate inference algorithm.

The objective of learning will be to maximize the likelihood of the data. Since during recognition we are interested in finding the compositions that best explain the image in terms of likelihood, this objective is well suited for learning. Not to overfit the data, however, we will instead use a likelihood with a penalty for the complexity of the representation. Further, due to the exponential complexity of the unsupervised structure learning problem, our strategy will be to learn one layer at a time, in a bottom-up fashion. Specifically, at each step, we will assume that layers from 1 to \( \ell - 1 \) have been learned and are held fixed during learning of layer \( \ell \). The problem thus reduces to learning a vocabulary at layer \( \ell \) by maximizing the penalized log-likelihood:

\[
L(D; \Omega^\ell, \Theta^\ell) = \sum_{k=1}^N \sum_{i} \log p(I_k(x^i_k), \Omega^\ell, \Theta^\ell) - \lambda^\ell |\Omega^\ell|
\]

\[
= \sum_{k=1}^N \sum_{i} \log \sum_{j} p(I_k(x^i_k), T_{ki}, \omega^i_k, \Omega^\ell, \Theta^\ell) - \lambda^\ell |\Omega^\ell|,
\]  

(17)

where \( \lambda^\ell \) is a layer-specific penalty for the complexity of the vocabulary. The first sum runs over all \( N \) training images (which are the training data \( D \)), while the second sum runs over all neighborhoods in an image. From here on, we will omit the second sum for simplicity and assume that each training sample is an image neighborhood of appropriate size. The second term is a penalty term, which corresponds to the sum of the number of parts across the compositions at layer \( \ell \): \( |\Omega^\ell| = \sum_{\omega^\ell} P_{\omega^\ell} \). The sum runs over all compositions at layer \( \ell \) of the vocabulary and \( P_{\omega^\ell} \) denotes the number of parts that \( \omega^\ell \) has. We will thus prefer structurally simpler compositions with high repeatability and descriptive power.

We will learn the vocabulary with a MCMC approach. Specifically, the objective in (17) will be used as a scoring function to compare different proposals for the vocabularies. At each step of the MCMC iteration, we will learn a proposal vocabulary at layer \( \ell \), use EM to learn the parameters and calculate the score of the learned vocabulary. Finally, the vocabulary with the highest score will be selected.

We will first assume that the structure of the representation is already known and show how to estimate the parameters using the EM algorithm. In Sec. 5.2 we propose how to learn also the structure of the vocabulary.

5.1 Learning the parameters

Let us for now assume that the structure (the number and the structure of the compositions) up to layer \( \ell \) is given. We can employ the EM algorithm to estimate the parameters for each of the compositions at layer \( \ell \) of the vocabulary. Since the model from one layer to the next is similar to the constellation model [53], we will adopt some of their derivations.

Instead of directly maximizing the likelihood in (17), EM iteratively re-estimates the parameters by maximizing the cost functions:

\[
Q(\hat{\Theta}^\ell|\Theta^\ell) = \sum_k E[ \log p(I_k(x^i_k), T_k, \omega^i_k|\hat{\Theta}^\ell) ]
\]

(18)

Following [53], differentiating \( Q \) with respect to each of the parameters and setting the derivatives to 0 gives the following updates for the parameters, which constitute the M step of the EM algorithm:

\[
\hat{\mu}_{\omega^i,j} = \frac{\sum_k p(\omega^i_k|I_k) E_{\omega^i,j}\langle q_k \rangle}{\sum_k p(\omega^i_k|I_k)}
\]

\[
\hat{\Sigma}_{\omega^i,j} = \frac{\sum_k p(\omega^i_k|I_k) E_{\omega^i,j}\langle q_k q_k^T \rangle}{\sum_k p(\omega^i_k|I_k)} - \hat{\mu}_{\omega^i,j} \hat{\mu}_{\omega^i,j}^T
\]

\[
\hat{\theta}_{\text{app}}^\ell(\omega^{\ell-1}) = \frac{\sum_k p(\omega^i_k|I_k) E_{\omega^{\ell-1}}\langle \delta_{\omega^{\ell-1},\omega^i_k} \rangle}{\sum_k p(\omega^i_k|I_k)}
\]

\[
\hat{p}(\omega^\ell) = \frac{1}{N} \sum_k p(I_k|\omega^\ell)
\]

where \( q_k = x^i_k - \hat{x}^i_k \) and \( \delta \) here denotes the Kronecker delta. The expectation \( E_{\omega^i,j}[\cdot] \) is taken with respect to the posterior density \( p(T_k|I_k, \omega^i_k, \hat{\Theta}^\ell) \).
In the E-step we need to compute the sufficient statistics \( E_{\omega_j}[q], E_{\omega_j}[q q^T] \) and the posterior density
\[
p(\omega^\ell | I_k) = \frac{p(I_k|\omega^\ell)p(\omega^\ell)}{\sum_i p(I_k|\omega^\ell)p(\omega^\ell)},
\]
where the likelihood \( p(I_k|\omega^\ell) \) is calculated as described in Sec. 4.2.

We only still need to infer the parameters for the first layer models in the vocabulary. This is done by estimating the parameters \((\mu_1^\ell, \Sigma_1^\ell)\) of a multivariate Gaussian distribution for each model \(\omega_1^\ell\): Each Gabor feature vector \(f\) is first normalized to have the dominant orientation equal to 1. All features \(f\) that have the \(i\)-th dimension equal to 1 are then used as the training examples for estimating the parameters \((\mu_1^\ell, \Sigma_1^\ell)\).

5.2 Learning the structure

Here we explain how we learn the proposal set of compositions, their structure and rough initializations of the parameters (which is important for initializing EM).

We will assume that for each training image \(I\) we have the inference graph \(G_I = (Z^1 \cup \cdots \cup Z^{\ell-1}, E)\) built up to layer \(\ell - 1\), which can be obtained as described in Sec. 4.2.

Our learning strategy will be the following:

1) First learn the geometry distributions between all possible pairs of compositions from layer \(\ell - 1\).
2) Detect the modes in these distributions which will define two-part compositions called duplets.
3) Find a set of compositions, where each composition is a set of (frequently) co-occurring duplets. The set (vocabulary) will be learned by the MCMC approach, i.e., by stochastically searching for the vocabulary with the optimal score (17).

The overall learning procedure is illustrated in Fig. 5.

5.2.1 Learning spatial correlations between parts

We commence by learning the spatial constraints among the pairs of compositions \((\omega_1^{\ell-1}, \omega_2^{\ell-1}) \in \Omega^{\ell-1} \times \Omega^{\ell-1}\). The first composition \(\omega_1^{\ell-1}\) in the pair plays the role of a reference. This means that the spatial constraints will be learned relative to it. We further limit the relative distances between the composition pairs to be at most \(r^\ell\) (the choice of \(r^\ell\) is described below). We learn the spatial constraints by means of two-dimensional histograms \(h_{ij}^\ell: [-r^\ell, r^\ell]^2 \to \mathbb{R}\).

During training, each histogram \(h_{ij}^\ell\) is updated at \(z^{\ell-1} - z^{\ell-1}\) for each pair of hidden states \((z^{\ell-1}, z^{\ell-1})\), where \(z^{\ell-1} = (\omega_1^{\ell-1}, x^{\ell-1})\) and \(z^{\ell-1} = (\omega_2^{\ell-1}, x^{\ell-1})\), satisfying:

1) \(|x^{\ell-1} - x^{\ell-1}| \leq r^\ell\), that is, \(z^{\ell-1}\) lies in the circular area around \(z\) with radius \(r^\ell\).
2) \(z^{\ell-1}\) and \(z^{\ell-1}\) have sufficiently disjoint supports.

This is due to our assumption that the compositions have a tree topology, i.e. none of the parts in each composition spatially overlap (or rather, the probability that two parts overlap is small). The overlap of the supports is calculated as \(o(z^{\ell-1}, z^{\ell-1}) = \frac{|\text{supp}(z^{\ell-1}) \cap \text{supp}(z^{\ell-1})|}{|\text{supp}(z^{\ell-1}) \cup \text{supp}(z^{\ell-1})|}\). We allow for a small overlap of the parts, \(o(z^{\ell-1}, z^{\ell-1}) < 0.2\).

The histograms are updated for all inference graphs \(G_I\) and all the admissible pairs of hypotheses at layer \(Z^{\ell-1}\).

The choice of \(r^\ell\). The correlation between the relative positions of the hypotheses is the highest at relatively small distances as depicted in Fig. 7. Depending on the factors of the spatial contraction, the radii \(r^\ell\) in this paper are set to 8 for layer 2, 12 for the higher layers, and 15 for the final, object layer. Note that the circular regions are defined at spatial resolution of layer \(\ell - 1\) which means that the area, if projected back to the original resolution, covers a much larger area in an image.

5.2.2 Learning compositions of parts

From the learned pairwise spatial distributions we form duplets via the distribution modes. Fig. 6 illustrates the clustering procedure. For each mode we estimate the mean \(\mu^\ell\) and variance \(\Sigma^\ell\) by fitting a Gaussian distribution around it. Each of these modes thus results in a two-part composition with initial parameters: \(P = 2, \theta_{app1}^\ell(\omega_1^\ell) = 1, \theta_{app2}^\ell(\omega_1^\ell) = 1\) and \(\theta_g^\ell = (\mu^\ell, \Sigma^\ell)\) (the first, reference part, in a composition is always assigned the default parameters as explained in Sec. 3). For the higher layers, \(\ell > 3\), where the co-occurrences are sparser, we additionally smooth the learned histograms prior to clustering, to regularize the data.

We can now perform inference with the complete set of duplets. Define a duplet co-occurrence for a particular training image neighborhood \(I_k\) as a set of duplets that best explain \(I_k\) (the likelihood of \(I_k\) defined by (11) is the highest for this particular set of duplets) and forms a tree. A duplet co-occurrence forms a composition and we will denote it by \(\omega^\ell\). During training, a list \(\Omega^\ell\) of all possible co-occurrences of duplets is kept. Note that for each composition in \(\Omega^\ell\) we also have the (rough) parameters of the distribution (previous paragraph), \(\Theta^\ell\).

Each \(\omega^\ell\) is additionally represented by a “count” value \(f(\omega^\ell)\) which is taken to be the value of the log-likelihood. Thus, in training, for each neighborhood \(I_k\), the count of the corresponding \(\omega^\ell\) is updated accordingly, \(f(\omega^\ell) = f(\omega^\ell) + \log q(I_k, \omega^\ell)\).

Among \(\Omega^\ell\) we need to select a set of compositions (a vocabulary) yielding the highest score in (17). We adopt a greedy approach and further refine it by the MCMC algorithm. Let \(\Omega^\ell_{g}\) denote the current set of compositions selected by the greedy algorithm. At each step of the iteration, we do the following: for each neighborhood \(I_k\) we update the count for the composition \(\omega^\ell \in \Omega^\ell_{g}\) that is most likely to have generated the neighborhood (has the highest likelihood), by the difference in its likelihood and the likelihood under the compositions from \(\omega^\ell \in \Omega^\ell \setminus \Omega^\ell_{g}\):
\[
f(\omega^\ell) = f(\omega^\ell) + \log q(I_k, \omega^\ell) - \max_{\omega^\ell \in \Omega^\ell \setminus \Omega^\ell_{g}} \log q(I_k, \omega^\ell).
\]

After accounting for all the neighborhoods, we select the composition with the highest count.
We next employ a stochastic MCMC algorithm to get the final vocabulary \( \Omega^\ell \). The first state of the Markov chain is the vocabulary \( \Omega_0^\ell \) obtained with the greedy selection. Let \( \Omega_t^\ell \) denote the vocabulary at the current state of the chain. At each step, we run the EM to also get the parameters \( \Theta_t^\ell \), where \( \Theta_t^\ell \) is always used to initialize EM. We either exchange/add/remove one composition from \( \Omega_t^\ell \) with another one from \( \Omega_t^\ell \setminus \Omega_t^\ell \) to get the vocabulary \( \Omega_{t+1}^\ell \). The vocabulary \( \Omega_{t+1}^\ell \) is accepted as the next state of the Markov chain with probability

\[
\min(1, \quad \beta^{L(D; \Theta_{t+1}^\ell, \Theta_t^\ell) - L(D; \Omega_{t+1}^\ell, \Theta_t^\ell)}), \quad \beta > 1
\]

(20)

according to the Metropolis-Hastings algorithm.

The vocabulary at layer \( \ell \) is finally defined as the \( \Omega^\ell \) with maximal value \( L(D; \Omega^\ell, \Theta^\ell) \), after running several iterations of the M-H algorithm. We usually perform 100 iterations.

To train the top, object-layer of compositions, we make use of the available supervision, similarly as in the constellation model [53]. During the greedy selection of the compositions, we always select the one with the highest detection accuracy (we use the F-measure).

### 5.3 Learning a multi-class vocabulary

We learn multiple object classes with the following strategy. Learning of the lower layers which are more generic is performed over the images of all classes (thus maximizing sharing between them). Moreover, we prefer to learn the lower layers from a set of natural images, which avoids knowing the classes in advance and results in a vocabulary which is more in tune to the statistics of natural images. Because the general, class-independent statistics of shapes becomes very sparse in the higher layers, we learn layers 4 and higher sequentially — by updating the vocabulary with the compositions learned for one class after another. When learning each class we use the vocabulary learned so far and only add those compositions to each of the layers which are still needed to represent the new object class.

When training the higher layers we use the bounding box information of the objects. We additionally scale the images so that the diagonal of the bounding box is approximately 250 pixels. The size of the training images has a direct influence on the number of layers being learned (due to compositionality, the number of layers needed to represent the whole shapes of the objects is logarithmic in the number of extracted edge features in the image). In order to learn a 6-layer hierarchy, the 250 pixel diagonal constraint has proven to be a good choice.

Once we have learned the multi-class vocabulary, we could, in principle, run the EM algorithm to re-learn the parameters over the complete hierarchy in a similar fashion as in the Hierarchical HMMs [50]. However, we have not done so in this paper.

#### 5.3.1 Learning the thresholds for the tests

Given that the object class representation is known, we can learn the thresholds \( \tau_{cd} \) to be used in our (approximate) inference algorithm (Sec. 4.2.3).

We use a similar idea to that of Amit et al. [39] and Fleuret and Geman [37]. The main goal is to learn the thresholds in way that nothing is lost with respect to the accuracy of detection, while at the same time optimizing for the efficiency of inference.

Specifically, by running the inference algorithm on the set of class training images \( \{I_k\} \), we obtain the
object class likelihoods \( \{q(I_k, c_k)\} \), the corresponding most probable hidden trees of states \( \{T(I_k, c_k)\} \) and their probabilities \( \{\delta(I_k, c_k)\} \). For each composition \( \omega^f \) in the vocabulary, we then define \( \tau_{\omega^f} \) as the minimal probability obtained in the training set:

\[
\tau_{\omega^f} = \min_k \min_{x^f = (\omega^f, x^f) \in T(I_k, c_k)} \delta(I_k(x^f), \omega^f)
\]  

(21)

For a better generalization, however, one can rather take a certain percentage of the value on the right [37].

6 Experimental results

The evaluation is separated into two parts. We first show the capability of our method to learn generic shape structures from natural images and apply them to an object classification task. Second, we utilize the approach for multi-class object detection on 15 object classes.

All experiments were performed on one core on an Intel Xeon-3 CPU 2.66 Ghz computer. The method is implemented in C++. Filtering is performed on CUDA.

6.1 Natural statistics and object classification

We applied our learning approach to a collection of 1500 natural images. Learning was performed only up to layer 3, with the complete learning process taking roughly 5 hours. The learned hierarchy consisted of 160 compositions on Layer 2 and 553 compositions on Layer 3. A few examples from both layers are depicted in Fig. 9 (note that the images shown are only the main shapes generated by the generative models). The learned features include corners, end-stopped lines, various curvatures, T- and L-junctions. Interestingly, these structures resemble the ones predicted by the Gestalt rules [20].

To put the proposed hierarchical framework in relation to other categorization approaches which focus primarily shape, the learned 3-layer vocabulary was tested on the Caltech-101 database [54]. The Caltech-101 dataset contains images of 101 different object categories with the additional background category. The number of images varies from 31 to 800 per category, with the average image size of roughly 300 \( \times \) 300 pixels.

Each image was processed at 3 scales spaced apart by \( \sqrt{2} \). The likelihoods of the hidden states were combined with a linear one-versus-all SVM classifier [55]. This was done as follows: for each image a vector of a dimension equal to the number of compositions at the particular layer was formed. Each dimension of the vector, which corresponds to a particular type of a composition (e.g. an L-junction) was obtained by summing over all the likelihoods of the hidden states coding this particular type of composition. To increase the discriminative information, a radial sampling was used similarly as in [56]: each image was split into 5 orientation bins and 2 distances from the image center, for which separate vectors (as described previously) were formed and consequently stacked together into one, high-dimensional vector.

The results, averaged over 8 random splits of train and test images, are reported in Table 1 with classification rates of existing hierarchical approaches shown for comparison. For 15 and 30 training images we obtained a 60.5\% and 66.5\% accuracy, respectively, which is slightly better than the most related hierarchical approaches [11], [57], [29], comparable to those using more sophisticated discriminative methods [58], [59], and slightly worse than those using additional information such as color or regions [60].

We further tested the classification performance by varying the number of training examples. For testing, 50 examples were used for the classes where this was possible and less otherwise. The classification rate was normalized accordingly. In all cases, the result was averaged over 8 random splits of the data. The results are presented and compared with existing methods in Fig. 8. We only compare to approaches that use shape information alone and not also color and texture. Overall, better performance has been obtained in [61].

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Average classification rate (in %) on Caltech-101.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( N_{\text{train}} = 15 )</td>
</tr>
<tr>
<td>Mutch et al. [57]</td>
<td>51</td>
</tr>
<tr>
<td>Ommer et al. [11]</td>
<td>/</td>
</tr>
<tr>
<td>Ahmed at al. [59]</td>
<td>58.1</td>
</tr>
<tr>
<td>Yu at al. [58]</td>
<td>59.2</td>
</tr>
<tr>
<td>Lee at al. [31]</td>
<td>57.7</td>
</tr>
<tr>
<td>Jarrett at al. [29]</td>
<td>/</td>
</tr>
<tr>
<td>our approach</td>
<td>60.5</td>
</tr>
</tbody>
</table>

Fig. 8. Classification performance on Caltech-101.

6.2 Object class detection

The approach was tested on 15 diverse object classes from standard recognition datasets. The basic information is given in Table 2. These datasets are known to be challenging because they contain a high amount of clutter, multiple objects per image, large scale differences of objects and exhibit a significant intra class variability.

For training we used the bounding box information of objects or the masks if they were available. Each
object was resized so that its diagonal in an image was approximately 250 pixels. For testing, we up-scaled each test image by a factor 3 and used 6 scales for object detection. When evaluating the detection performance, a detection is counted as correct, if the predicted bounding box $b_{fg}$ coincides with the ground truth $b_{gt}$ more than 50%: $\frac{|\text{area}(b_{fg} \cap b_{gt})|}{\text{area}(b_{fg} \cup b_{gt})} > 0.5$. On the ETH dataset and INRIA horses this threshold is lowered to 0.3 to enable a fair comparison with the related work [44]. The performance is given either with recall at equal error rate (EER) or positive detection rate at low FPPI, depending on the type of results reported on these datasets thus-far.

6.2.1 Single class learning and performance

We first evaluated our approach to learn and detect each object class individually. We report the training and inference times, and the accuracy of detection, which will then be compared to the multi-class case in Subsec. 6.2.2 to test the scalability of the approach.

Training time. To train a class it takes on average $20-25$ minutes. For example, it takes $23$ minutes to train on Apple logo, $31$ for giraffe, $17$ for swan, $25$ for cow, and $35$ for the horse class. For comparison, in Shotton et al. [4], training on 50 horses takes roughly $2$ hours (on a $2.2$ GHz machine using a C# implementation).

Inference time. Detection for each individual class takes from $2-4$ seconds per image, depending on the size of the image and the amount of texture it contains. Other related approaches report approx. 5 to 10 times higher times (in seconds): [35]: $20-30$, [38]: $16.9$, [62]: $20$, [1]: $12-18$, however, at slightly older hardware.

Detection performance. The ETH experiments are performed in a 5-fold cross-validation obtained by sampling 5 subsets of half the class images at random. The test set for evaluating detection consists of all the remaining images in the dataset. The detection performance is given as the detection rate at the rate of 0.4 false-positives per image (FPPI), averaged over the 5 trials as in [44]. The detection performance is reported in Table 3. Similarly, the results for the INRIA horses are given in a 5-fold cross-validation obtained by sampling 5 subsets of 50 class images at random and using the remaining 120 for testing. The test set also includes 170 negative images to allow for a higher FPPI rate. With respect to [44], we achieve a better performance for all classes, most notably for giraffes (24.7%). Our method performs comparably to a discriminative framework by Fritz and Schiele [62]. Better performances have recently been obtained by Maji and Malik [63] (93.2%) and Ommer and Malik [52] (88.8%) using Hough voting and SVM-based verification.

To the best of our knowledge, no other hierarchical approach has been tested on this dataset so far.

For the experiments on the rest of the datasets we report the recall at EER. The results are given in Table 3. Our approach achieves competitive detection rates with respect to the state-of-the-art. Note also that [34], [6] used 150 training examples of motorbikes, while we only used 50 (to enable a fair comparison with [3] on GRAZ).

While the performance in a single-class case is comparable to the current state-of-the-art, the main advantage of our approach are its computational properties when the number of classes is higher, as demonstrated next.

6.2.2 Multi-class learning and performance

To evaluate the scaling behavior of our approach we have incrementally learned 15 classes one after another.

The learned vocabulary. A few examples of the learned shapes at layers 4 to 6 are shown in Fig.10 (only samples from the generative model are depicted). An example of a complete object-layer composition is depicted in Fig. 14. It can be seen that the approach has learned the essential structure of the class well, not missing any important shape information.

Degree of composition sharing among classes. To see how shared are the vocabulary compositions between classes of different degrees of similarity, we depict the learned vocabularies for two visually similar classes (motorbike and bicycle), two semi-similar classes (giraffe and horse), and two dissimilar classes (swan and car_front) in Fig. 11. The nodes correspond to compositions and the links denote the compositional relations between them and their parts. The green nodes represent the compositions used by both classes, while the specific colors denote class specific compositions. If a spatial relation is shared as well, the edge is also colored green. The computational advantage of our hierarchical representation over flat ones [3], [6], [2] is that the compositions are shared at several layers of the vocabulary, which significantly reduces the overall complexity of inference. Even for the visually dissimilar classes, which may not have any complex parts in common, our representation is highly shared at the lower layers of the hierarchy.

To evaluate the shareability of the learned compositions among the classes, we use the following measure:

$$\text{deg}_\text{share}(\ell) = \frac{1}{|\Omega|} \sum_{\omega \in \Omega^\ell} \left( \frac{\# \text{ of classes that use } \omega^\ell - 1}{\# \text{ of all classes}} \right),$$

defined for each layer $\ell$ separately. By “$\omega^\ell$ used by class $c$” it is meant that the probability of $\omega^\ell$ under $c$ is not equal to zero. To give some intuition behind the measure: $\text{deg}_\text{share} = 0$ if no composition from layer $\ell$ is shared (each class uses its own set of compositions), and it is 1 if each composition is used by all the classes. Fig. 13 (b) plots the values for the learned 15-class representation. Beside the mean (which defines $\text{deg}_\text{share}$), the plots also show the standard deviation.

The size of the vocabulary. It is important to test how the size of the vocabulary (the number of compositions at each layer) scales with the number of classes, since this has a direct influence on inference time. We report the size as a function of the number of learned classes in Fig. 12. For the “worst case” we take the independent training approach: a vocabulary for each class is learned independently of other classes and we sum over the sizes of these separate vocabularies. One can observe
a logarithmic tendency especially at the lower layers. This is particularly important because the complexity of inference is much higher for these layers (because they contain less discriminative compositions which are detected more numerously in images). Although the fifth layer contains highly class specific compositions, one can still observe a logarithmic increase in size. The final, object layer, is naturally linear in the number of classes, but it does, however, learn less object models than is the number of all training examples of objects in each class.

We further compare the scaling tendency of our approach with the one reported for a non-hierarchical representation by Opelt et al. [3]. The comparison is given in Fig. 13 (a) where the worst case is taken as in [3]. We compare the size of the class-specific vocabulary at layer 5, where the learned compositions are of approximately the same granularity as the features used in [3]. We additionally compare the overall size of the vocabulary (a sum of the sizes over all layers). Our approach achieves a substantially better scaling tendency, which is due to sharing at multiple layers of representation. This, on the one hand, compresses the overall representation, while it also attains a higher variability of the compositions in the higher layers and consequently, a lower number of them are needed to represent the classes.

Fig. 13 (d) shows the storage demands as a function of the number of learned classes. This is the actual size of the vocabulary stored on a hard disk. Notice that the 15-class hierarchy takes only 1.6Mb on disk. 

Inference time. We further test how the complexity of inference increases with each additional class learned. We randomly sample ten images per class and report the detection times averaged over all selected images. The times are reported as a function of the number of learned classes. The results are plotted in Fig. 13 (c), showing that the running times are significantly faster than the “worst case” (the case of independent class representation, in which each separate class vocabulary is used to detect objects). It is worth noting, that it takes only 16 seconds per image to apply a vocabulary of all 15 classes.

Detection performance. We additionally test the multi-class detection performance. The results are presented in Table 4 and compared with the performance of the single class vocabularies. The evaluation was the following. For the single class case, a separate vocabulary was trained on each class and evaluated for detection independently of other classes. For the multi-class case, a joint multi-class vocabulary was learned (as explained in Sec. 6.2.2) and detection was performed as proposed in Sec. 5.3, that is, we allowed for competition among hypotheses explaining partly the same regions in an image.

Overall, the detection rates are slightly worse in the multi-class case, although in some cases the multi-class case outperformed the independent case. The main reason for a slightly reduced performance is due to the fact that the representation is generative and is not trained to discriminate between the classes. Consequently, it does not separate similar classes sufficiently well and a wrong hypothesis may end up inhibiting the correct one. As part of the future work, we plan to also incorporate more discriminative information into the representation and increase the recognition accuracy in the multi-class case. On the other hand, the increased performance for the multi-class case for some objects is due to feature sharing among the classes, which results in better regularization and generalization of the learned representation.

7 Summary and Conclusions

We proposed a novel approach which learns a hierarchical compositional shape vocabulary to represent multiple object classes in an unsupervised manner. Learning is performed bottom-up, from small oriented contour fragments to whole-object class shapes. The vocabulary is learned recursively, where the compositions at each layer are combined via spatial relations to form larger and more complex shape compositions.

Experimental evaluation was two-fold: one that shows the capability of the method to learn generic shape structures from natural images and uses them for object classification, and another one that utilizes the approach in multi-class object detection. We have demonstrated a competitive classification and detection performance, fast inference even for a single class, and most importantly, a logarithmic growth in the size of the vocabulary (at least in the lower layers) and, consequently, scalability of inference complexity as the number of modeled classes grows. The observed scaling tendency of our hierarchical framework goes well beyond that of a flat approach [3]. This provides an important showcase that highlights learned hierarchical compositional vocabularies as a suitable form of representing a higher number of object classes.

8 Future work

There are numerous directions for future work. One important aspect would be to include multiple modalities in the representation. Since many object classes have distinctive textures and color, adding this information to our model would increase the range of classes that the method could be applied to. Additionally, modeling texture could also boost the performance of our current model: since textured regions in an image usually have a lot of noisy feature detections, they are currently more susceptible to false positive object detections. Having a model of texture could be used to remove such regions from the current inference algorithm.

Part of our ongoing work is to make the approach scalable to a large number of object classes. To achieve this, several improvements and extensions are still needed. We will need to incorporate the discriminative information into the model (to distinguish better between similar classes), make use of contextual information to improve performance in the case of ambiguous information (small objects, large occlusion, etc) and use attention mechanisms to speed-up detection in large complex
Fig. 9. Examples of learned vocabulary compositions (with the exception of a fixed Layer 1) learned on 1500 natural images. Only the mean of the compositions are depicted.

![Layer 1](image1.png)

![Layer 2](image2.png)

![Layer 3](image3.png)

![Layer 4](image4.png)

![Layer 5](image5.png)

![Layer 6 (object layer O)](image6.png)

Layer 6 (object layer O)

- Apple logo
- person
- bottle
- mug
- swan
- cup
- cow
- horse
- car
- bicycle
- face
- giraffe

Fig. 10. Examples of samples from compositions at layers 4, 5, and top, object layer learned on 15 classes.

**TABLE 2**

<table>
<thead>
<tr>
<th>dataset</th>
<th>ETH shape</th>
<th>UIUC</th>
<th>Weizmann</th>
<th>INRIA</th>
<th>TUD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>class</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Apple logo</td>
<td>bottle</td>
<td>giraffe</td>
<td>mug</td>
<td>swan</td>
</tr>
<tr>
<td># train im.</td>
<td>19</td>
<td>21</td>
<td>30</td>
<td>24</td>
<td>15</td>
</tr>
<tr>
<td># test im.</td>
<td>21</td>
<td>27</td>
<td>57</td>
<td>24</td>
<td>17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dataset</th>
<th>GRAZ</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>class</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bicycle_side</td>
<td>bottle</td>
<td>car_front</td>
<td>car_rear</td>
<td>cow</td>
</tr>
<tr>
<td># train im.</td>
<td>45</td>
<td>/</td>
<td>20</td>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td># test im.</td>
<td>53</td>
<td>64</td>
<td>20</td>
<td>400</td>
<td>65</td>
</tr>
</tbody>
</table>

images. Furthermore, a taxonomic organization of object classes could further improve the speed of detection and, possibly, also the recognition rates of the approach.
TABLE 3
Detection results. On the ETH shape and INRIA horses we report the detection-rate (in %) at 0.4 FPPI averaged over five random splits train/test data. For all the other datasets the results are reported as recall at equal-error-rate (EER).

<table>
<thead>
<tr>
<th>ETH shape</th>
<th>class</th>
<th>[44]</th>
<th>[62]</th>
<th>our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>applelogo</td>
<td>83.2(1.7)</td>
<td>89.9(4.5)</td>
<td>87.3(2.6)</td>
</tr>
<tr>
<td></td>
<td>bottle</td>
<td>83.2(7.5)</td>
<td>76.8(6.1)</td>
<td><strong>86.2(2.8)</strong></td>
</tr>
<tr>
<td></td>
<td>giraffe</td>
<td>58.6(14.6)</td>
<td>90.5(5.4)</td>
<td>83.3(4.3)</td>
</tr>
<tr>
<td></td>
<td>mug</td>
<td>83.0(8.0)</td>
<td>82.7(5.1)</td>
<td><strong>84.6(2.3)</strong></td>
</tr>
<tr>
<td></td>
<td>swan</td>
<td>75.4(13.4)</td>
<td>84.0(8.4)</td>
<td>78.2(5.4)</td>
</tr>
<tr>
<td></td>
<td>average</td>
<td>76.8</td>
<td>84.8</td>
<td>83.7</td>
</tr>
<tr>
<td>INRIA</td>
<td>horse</td>
<td>84.8(2.6)</td>
<td></td>
<td><strong>85.1(2.2)</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UIUC</th>
<th>class</th>
<th>related work</th>
<th>our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>car_side, multiscale</td>
<td>90.6 [57]</td>
<td>93.5</td>
</tr>
<tr>
<td>Weizmann</td>
<td>horse_multiscale</td>
<td>89.0 [4]</td>
<td>93.0</td>
</tr>
<tr>
<td>TUD</td>
<td>motorbike</td>
<td>87 [6]</td>
<td>88 [34]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GRAZ</th>
<th>class</th>
<th>[3]</th>
<th>[4]</th>
<th>our approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>face</td>
<td>96.4</td>
<td>97.2</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>bicycle_side</td>
<td>72</td>
<td>67.9</td>
<td>68.5</td>
</tr>
<tr>
<td></td>
<td>bottle</td>
<td>91</td>
<td>90.6</td>
<td>89.1</td>
</tr>
<tr>
<td></td>
<td>cow</td>
<td>90</td>
<td>98.5</td>
<td>96.9</td>
</tr>
<tr>
<td></td>
<td>cup</td>
<td>91.2</td>
<td>95</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>car_front</td>
<td>90</td>
<td>70.6</td>
<td>76.5</td>
</tr>
<tr>
<td></td>
<td>car_rear</td>
<td>97.7</td>
<td>98.2</td>
<td>97.5</td>
</tr>
<tr>
<td></td>
<td>horse_side</td>
<td>91.8</td>
<td>93.7</td>
<td>93.7</td>
</tr>
<tr>
<td></td>
<td>motorbike</td>
<td>95.6</td>
<td>99.7</td>
<td>93.0</td>
</tr>
<tr>
<td></td>
<td>mug</td>
<td>99.3</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>person</td>
<td>92.6</td>
<td>52.4</td>
<td><strong>60.4</strong></td>
</tr>
</tbody>
</table>

Fig. 11. Sharing of compositions between classes. Each plot shows a vocabulary learned for two object classes. The bottom nodes in each plot represent the 6 edge orientations, one row up denotes the learned second layer compositions, etc. The sixth row (from bottom to top) are the whole-shape, object-layer compositions (i.e., different aspects of objects in the classes), while the top layer is the class layer which pools the corresponding object compositions together. The green nodes denote the shared compositions, while specific colors show class-specific compositions. From left to right: Vocabularies for: (a) two visually similar classes (motorbike and bicycle), (b) two semi-similar object classes (giraffe and horse), (c) two visually dissimilar classes (swan and car_front). Notice that even for dissimilar object classes, the compositions from the lower layers are shared between them.

Fig. 12. Size of representation (the number of compositions per layer) as a function of the number of learned classes. “Worst case” denotes the sum over the sizes of single-class vocabularies.

Fig. 13. From left to right: (a) A comparison in scaling to multiple classes of the approach by Opelt et al. [3] (flat) and our hierarchical approach; (b) Degree of sharing for the multi-class vocabulary; (c) Average inference time per image as a function of the number of learned classes; (d) Storage (size of the hierarchy stored on disk) as a function of the number of learned classes.
TABLE 4
Comparison in detection accuracy for single- and multi-class object representations and detection procedures.

<table>
<thead>
<tr>
<th>class</th>
<th>Apple</th>
<th>bottle</th>
<th>giraffe</th>
<th>mug</th>
<th>swan</th>
<th>horse</th>
<th>cow</th>
<th>mbike</th>
<th>bicycle</th>
<th>car_front</th>
<th>car_side</th>
<th>car_rear</th>
<th>face</th>
<th>person</th>
<th>cup</th>
</tr>
</thead>
<tbody>
<tr>
<td>measure</td>
<td>detection rate (in %) at 0.4 FPPI</td>
<td>recall (in %) at EER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>single</td>
<td>88.6</td>
<td>85.5</td>
<td>83.5</td>
<td>84.9</td>
<td>75.8</td>
<td>84.5</td>
<td>96.9</td>
<td>83.2</td>
<td>68.5</td>
<td>76.5</td>
<td>97.5</td>
<td>93.5</td>
<td>94.0</td>
<td>60.4</td>
<td>85.0</td>
</tr>
<tr>
<td>multi</td>
<td>84.1</td>
<td>80.0</td>
<td>80.2</td>
<td>80.3</td>
<td>72.7</td>
<td>82.3</td>
<td>95.4</td>
<td>83.2</td>
<td>64.8</td>
<td>82.4</td>
<td>93.8</td>
<td>92.0</td>
<td>93.0</td>
<td>62.5</td>
<td>85.0</td>
</tr>
</tbody>
</table>

Fig. 14. One learned object-layer composition $\omega^O$ for (a) bottle, (b) giraffe, and (c) bicycle, with the complete model structure shown. The blue patches denote the limiting circular regions, the color regions inside them depict the spatial relations (Gaussians). The nodes are the compositions in the vocabulary. Note that each composition models the distribution over the compositions from the previous layer for the appearance of its parts. For clarity, only the most likely composition for the part is decomposed further.

Fig. 15. Examples of detections. The links show the most probable hidden state activation for a particular (top-layer) class detection. The links are color-coded to denote different classes.

ACKNOWLEDGMENTS

This research has been supported in part by the following funds: Research program Computer Vision P2-0214 (Slovenian Ministry of Higher Education, Science and Technology) and EU FP7-215843 project POETICON.

REFERENCES


