Journal of Machine Learning Research () Submitted ; Published

# Complexity of Representation and Inference in Compositional Models with Part Sharing

**Alan Yuille** YUILLE@STAT.UCLA.EDU
*Department of Statistics*
*University of California*
*Los Angeles, CA 90095-1554 USA*

**Roozbeh Mottaghi** ROOZBEHM@ALLENAI.ORG
*Allen Institute for Artificial Intelligence*
*Seattle WA 98103, USA*

**Editor:**

## Abstract

This paper performs a complexity analysis of a class of serial and parallel compositional models of multiple objects and shows that they enable efficient representation and rapid inference. Compositional models are generative and represent objects in a hierarchically distributed manner in terms of parts and subparts, which are constructed recursively by part-subpart compositions. Parts are represented more coarsely at higher level of the hierarchy, so that the upper levels give coarse summary descriptions (e.g., there is a horse in the image) while the lower levels represents the details (e.g., the positions of the legs of the horse). This hierarchically distributed representation obeys the *executive summary* principle, meaning that a high level executive only requires a coarse summary description and can, if necessary, get more details by consulting lower level executives. The parts and subparts are organized in terms of hierarchical dictionaries which enables *part sharing* between different objects allowing efficient representation of many objects. The first main contribution of this paper is to show that compositional models can be mapped onto a parallel visual architecture similar to that used by bio-inspired visual models such as deep convolutional networks but more explicit in terms of representation, hence enabling part detection as well as object detection, and suitable for complexity analysis. Inference algorithms can be run on this architecture to exploit the gains caused by part sharing and executive summary. Effectively, this compositional architecture enables us to perform exact inference simultaneously over a large class of generative models of objects. The second contribution is an analysis of the complexity of compositional models in terms of computation time (for serial computers) and numbers of nodes (e.g., "neurons") for parallel computers. In particular, we compute the complexity gains by part sharing and executive summary and their dependence on how the dictionary scales with the level of the hierarchy. We explore three regimes of scaling behavior where the dictionary size (i) increases exponentially with the level of the hierarchy, (ii) is determined by an unsupervised compositional learning algorithm applied to real data, (iii) decreases exponentially with scale. This analysis shows that in some regimes the use of shared parts enables algorithms which can perform inference in time linear in the number of levels for an exponential number of objects. In other regimes part sharing has little advantage for serial computers but can enable linear processing on parallel computers.

**Keywords:** Compositional Models, Object Detection, Hierarchical Architectures, Part Sharing

© Alan Yuille and Roozbeh Mottaghi.

## 1. Introduction

A fundamental problem of vision is how to deal with the enormous complexity of images and visual scenes. The total number of possible images is astronomically large Kersten (1987). The number of objects is also huge and has been estimated at around 30,000 Biederman (1987). We observe also that humans do not simply detect objects but also *parse* them into their parts and subparts, which adds another level of complexity. How can a biological, or artificial, vision system deal with this complexity? For example, considering the enormous input space of images and output space of objects, how can humans interpret images in less than 150 msec Thorpe et al. (1996)?

There are three main issues involved. Firstly, how can a visual system be designed so that it can efficiently *represent* large numbers of objects, including their parts and subparts? Secondly, how can a visual system rapidly *infer* which object, or objects, are present in an input image and perform related tasks such as estimating the positions of their parts and subparts? And, thirdly, how can this representation be *learnt* in an unsupervised, or weakly supervised fashion? In summary, what visual *architectures* enable us to overcome these three issues?

Studies of mammalian visual systems offer some suggestions for how these complexity issues can be addressed. These visual systems are organized hierarchically with the lower levels (e.g., in areas V1 and V2) tuned to small image features while the higher levels (i.e. in area IT) are tuned to objects. This leads to a class of bio-inspired visual architectures, of varying degrees of biological plausibility, which represent the visual world in terms of hierarchies of features which are shared between multiple objects Fukushima (1988); Riesenhuber and Poggio (1999); Hinton et al. (2006); Serre et al. (2007); Adams and Williams (2003); Poon and Domingos (2010); Zeiler et al. (2010); LeCun and Bengio (1995); Borenstein and Ullman (2002); Kokkinos and Yuille (2011); Krizhevsky et al. (2012). This sharing of hierarchical features suggest ways to deal with the complexity issues but, to the best of our knowledge, there have been no complexity studies of these classes of architectures. We note that the purpose of this paper is not to attempt to model the known structure of mammalian visual systems. Instead, own goal of this work is to see whether it is possible to derive properties of mammalian visual systems from first principles by developing a visual architecture that addresses the complexity problems of vision.

In this paper, we address the complexity issues from the perspective of compositional models Geman et al. (2002) by extending and analyzing a specific class of models Zhu et al. (2008, 2010). Compositional models are cousins of the bio-inspired hierarchical models and we briefly discuss their similarities and differences in section (6). The key idea of compositionality is to represent objects by recursive composition from parts and subparts which enables: (i) part sharing between different objects, and (ii) hierarchically distributed representations of objects where the positions of parts are represented more coarsely than the position of subparts, which we call the executive-summary principle. This gives rise to natural learning and inference algorithms which proceed from sub-parts to parts to objects (e.g., inference is efficient because a leg detector can be used for detecting the legs of cows, horses, and yaks). The explicitness of the object representations helps quantify the efficiency of part-sharing, and executive summary, and make mathematical analysis possible. They also enable us to parse the objects into their parts and subparts, instead of only detecting objects. The compositional models we study are based on prior work Zhu et al. (2008, 2010) but we make several technical modifications. These include re-formulating the models so that the positions of the parts are restricted to lie on a set of discrete lattices (coarser lattices for high-level parts). This enables a parallel implementation of compositional models which helps clarify the

similarities, and differences, to bio-inspired models. For completeness, we briefly summarize how compositional models can be learnt in an unsupervised manner Zhu et al. (2008, 2010) and discuss how this relates to the memorization algorithms of Valiant (2000). But we emphasize that this paper does not directly address learning but instead explores the consequence of the representations which were learnt. We note that previous work has addressed the complexity of visual search and attention Blanchard and Geman (2005), Tsotsos (2011).

Our analysis assumes that objects are represented by hierarchical graphical probability models which are composed from more elementary models by *part-subpart compositions*. An object – a graphical model with $\mathcal{H}$ levels – is defined as a composition of $r$ parts which are graphical models with $\mathcal{H}-1$ levels. These parts are defined recursively in terms of subparts which are represented by graphical models of increasingly lower levels. It is convenient to specify these compositional models in terms of a set of dictionaries $\{\mathcal{M}_h : h = 0, ..., \mathcal{H}\}$ where the level-$h$ parts in dictionary $\mathcal{M}_h$ are composed in terms of $r$ level-$(h-1)$ parts in dictionary $\mathcal{M}_{h-1}$. The highest level dictionaries $\mathcal{M}_{\mathcal{H}}$ represent the set of all objects. The lowest level dictionaries $\mathcal{M}_0$ represent the elementary features that can be measured from the input image. Part-subpart composition enables us to construct a very large number of objects by different compositions of elements from the lowest-level dictionary (like building an object from a lego kit). It enables us to perform *part-sharing* during learning and inference, which can lead to enormous reductions in complexity, as our mathematical analysis will show. We stress that the parts and subparts are deformable (i.e. non-rigid).

There are three factors which enable computational efficiency. The first is *part-sharing*, as described above, which means that we only need to perform inference on the dictionary elements. The second is the *executive-summary principle*. This principle allows us to represent the state of a part coarsely because we are also representing the state of its subparts (e.g., a top level executive of a business company will only want to know that "there is a horse in the field" and will not care about the exact positions of its legs, which will be known by the lower level executives). For example, consider a letter $T$ which is composed of a horizontal and vertical bar. If the positions of these two bars are specified precisely, then we only need to represent the position of the letter $T$ more crudely (it is sufficient for it to be "bound" to the two bars, and for their positions to be represented separately). The third factor is *parallelism* which arises because the part dictionaries can be implemented in parallel. This is roughly similar to having a set of non-linear receptive fields for each dictionary element. This enables extremely rapid inference at the cost of a larger, but parallel, graphical model. The inference proceeds in a single bottom-up and top-down pass, where the bottom-up pass rapidly estimates an executive summary which enables the the top-down pass to estimate the lower-level details.

The compositional section (2) introduces the key ideas of compositional models and section (3) describes how they are used to generate images. Section (4) describes the compositional architecture and the inference algorithms for serial and parallel implementations. Section (5) performs a complexity analysis and shows potential exponential gains by using compositional models. Section (6) discusses the relations between compositional models and bio-inspired hierarchical models, briefly discusses how compositional models can be learnt in an unsupervised manner, discusses robust variants of compositional models and summaries an analysis of robustness (which is presented in the appendix).

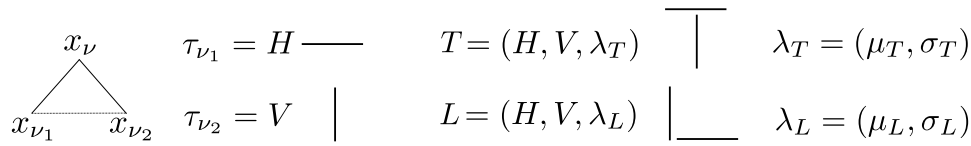



Figure 1: Compositional part-subpart models $T = (H, V, \lambda_T)$ and $L = (H, V, \lambda_L)$ for letters $T$ and $L$ are constructed from the same elementary components $\tau_H, \tau_V$, horizontal and vertical bar using different spatial relations $\lambda_T = (\mu_T, \sigma_T)$ and $\lambda_L = (\mu_L, \sigma_L)$, where $\mu$ denotes average displacement between the bars and $\sigma^2$ is the variance. The state $x_\nu$ of the parent node gives the summary position of the object, the *executive summary*, while the positions $x_{\nu_1}$ and $x_{\nu_2}$ of the components give the detailed positions of its components.

## 2. Compositional Models

Compositional models are based on the idea that objects are built by compositions of parts which, in turn, are compositions of more elementary parts. Section (2.1) describes part-subparts compositions which are the basic building block of our approach. In section (2.2) we discuss how to build objects by recursively making part-subpart compositions. Section (2.3) describes the state variables and how they take values on a hierarchy of lattices. In section (2.4) we discuss hierarchical dictionaries and part sharing. In section (2.5) we specify the probability distributions for objects.

### 2.1 Compositional Part-Subparts

We formulate part-subpart compositions by probabilistic graphical models which specify how a part is composed of its subparts. This consists of a parent node $\nu$ with a *type* $\tau_\nu$ and a *state variable* $x_\nu$. The parent node represents the part and has $r$ child nodes $Ch(\nu) = (\nu_1, ..., \nu_r)$, representing the subparts, which have types $\tau_{Ch(\nu)} = (\tau_{\nu_1}, ..., \tau_{\nu_r})$ and state variables $x_{Ch(\nu)} = (x_{\nu_1}, ..., x_{\nu_r})$ ($r$ is fixed for all part-subpart compositions in this paper). For concreteness the state variables represent the position of the parts, or subparts in the image (unless otherwise specified, our analysis is directly extended to other cases where they represent other image properties, such as mini-epitomes or active patches Papandreou et al. (2014); Mao et al. (2014)). Similarly, the types represent geometric entities (e.g., the letters $T, L$ or vertical/horizontal bars indicted by $V, H$). The type $\tau_\nu$ of the parent node is specified by the types $\tau_{\nu_i}$ of the child nodes and by the *spatial relations* $\lambda_\nu$ between the child nodes. Hence we express $\tau_\nu = (\tau_{\nu_1}, ..., \tau_{\nu_r}, \lambda_\nu)$. For example, in Figure (1) we represent the letters $T$ and $L$ by part-subparts where the parent has type "T" or "L". In both cases, there are two child nodes (i.e. $r = 2$) with types "vertical bar" and "horizontal bar" and the spatial relations are represented by $\lambda_T$ and $\lambda_L$ respectively. This is illustrated in Figure (1).

The spatial configuration of the part-subpart is represented by the state of the parent $x_\nu$ and the states of the children $x_{Ch(\nu)}$. The state of the parent $x_\nu$ provides an *executive summary* description of the part (e.g., there is a cow in the right side of the image) while the states of the children $x_{Ch(\nu)}$ gives the details (e.g., the positions of the head, torso, and legs of the cow). In this paper, we restrict the spatial position $x_\nu$ to take a discrete set of values, see section (2.3), which differs from earlier work where they were allowed to be continuous Zhu et al. (2008, 2010).

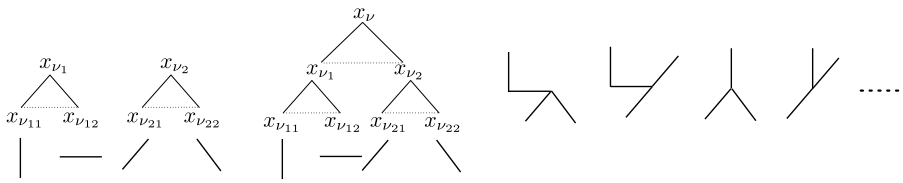



Figure 2: Object models are built recursively by part-subpart relations, Each object is represented by a hierarchical graph whose edges specify the part-subpart relationships. Each node $\nu$ has a type $\tau_\nu$ and a state variable $x_\nu$. Left Panel: Two part-subpart models, where the subparts are bars at four different angles (horizontal, vertical, forty-five degrees up, and forty-five degrees down). Center Panel: Combining two part-subpart models by composition to make a higher level model. Right Panel: Some examples of the higher-level models that can be obtained by different compositions.

The probability distribution for the part-subpart model relates the states of the part and the subparts by:

$$P(x_{Ch(\nu)}|x_\nu;\tau_\nu) = \delta(x_\nu - f(x_{Ch(\nu)}))h(x_{Ch(\nu)};\lambda_\nu). \tag{1}$$

Here $f(.)$ is a deterministic function, so the state of the parent node is determined uniquely by the state of the child nodes ($\delta(.)$ is a Dirac delta function if $x_\nu$ takes continuous values, and a Kronecker delta if $x_\nu$ takes discrete values). The function $h(.)$, which is parameterized by $\lambda$, specifies a distribution on the relative states of the child nodes. The distribution $P(x_{Ch(\nu)}|x_\nu;\tau_\nu)$ is local in the sense that $P(x_{Ch(\nu)}|x_\nu;\tau_\nu) = 0$, unless $|x_{\nu_i} - x_\nu|$ is smaller than a threshold for all $i = 1,..,r$. This requirement captures the intuition that subparts of a part are typically close together.

For example, in the simple example shown in Figure (1), the function $f(x_{\nu_1}, x_{\nu_2})$ specifies the average positions $x_{\nu_1}$ and $x_{\nu_2}$ of the horizontal and vertical bars, rounded off to the nearest position on the lattice. We set $h(.;\lambda)$ to be a discretized variant of the Gaussian distribution on the relative positions of the two bars, so $\lambda = (\mu, \sigma)$ where $\mu$ is the mean relative positions and $\sigma$ is the covariance. Observe that the parts $T$ and $L$ have the same subparts – horizontal and vertical bars – but they have different spatial relations $\lambda_T$ and $\lambda_L$ (because the mean relative positions of the bars are different for $T$'s and $L$'s). Hence the two compositional models for the $T$ and $L$ have types $T = (H, V, \lambda_T)$ and $L = (H, V, \lambda_L)$.

### 2.2 Representing an object using a hierarchical graph

We model an object recursively from parts and subparts using part-subpart compositions. This is illustrated in Figure (2) where we combine $T$'s and $L$'s with other parts to form more complex objects. The basic idea is to build increasingly complex shapes by treating the parts as subparts of higher order parts.

More formally, an object is represented by a graph with nodes $\mathcal{V}$. State variables $x_\nu$ and types $\tau_\nu$ are defined at the nodes $\nu \in \mathcal{V}$. The graph has a hierarchical structure with levels $h \in \{0, ..., \mathcal{H}\}$, where $\mathcal{V} = \bigcup_{h=0}^{\mathcal{H}} \mathcal{V}_h$. The edges in the graph specify the part-subpart compositions by connecting each node at level-$h$ to $r$ nodes at level-$h-1$. Each object has a single, root node, at level-$\mathcal{H}$. Any node $\nu \in \mathcal{V}_h$ (for $h > 0$) has $r$ children nodes $Ch(\nu)$ in $\mathcal{V}_{h-1}$ indexed by $(\nu_1, ..., \nu_r)$ (corresponding to the subparts of the part). Hence there are $r^{\mathcal{H}-h}$ nodes at level-$h$ (i.e. $|\mathcal{V}_h| = r^{\mathcal{H}-h}$).

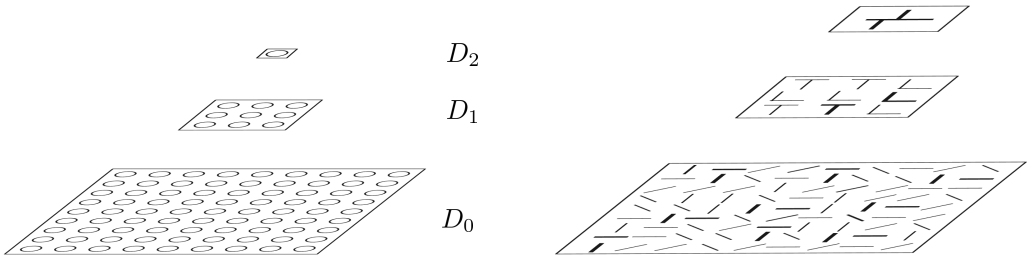



Figure 3: The hierarchical lattices. Left Panel: the size of the lattices decrease with scale by a factor $q = 1/9$ which helps enforce executive summary and prevent having multiple hypotheses which overlap too much. Right Panel: How an object is represented by a hierarchically distributed representation with coarse summary of position at the top level and more details at the lower levels.

The type of object is specified by the type $\tau_{\mathcal{H}}$ of the root node. Hence we use $\mathcal{V}^{\tau_{\mathcal{H}}}$ to refer to the nodes of the graph for object $\tau_{\mathcal{H}}$. The types of its descendant nodes are specified recursively by $\tau_{Ch(\nu)} = (\tau_{\nu_1}, ..., \tau_{\nu_r})$. The types of the lowest-level nodes, at $h = 0$ specify elementary features (e.g., horizontal and vertical bars). Hence we can think of an object as being constructed recursively by combining elementary structures with spatial relations (e.g., like building an object from a lego kit).

### 2.3 State Variables and Hierarchical Lattices

We require the state variables $\{x_\nu : \nu \in \mathcal{V}\}$ to satisfy the executive summary principle. To enforce this, we require that the positions of the subparts are specified at greater resolution than the positions of the parts. This is done by requiring that the state variables take values on a hierarchical lattice which we describe in this section. More precisely, the positions of the parts and subparts are specified by state variables $x_\nu$ which take values in a *set of lattices* $\{\mathcal{D}_h : h = 0, ..., \mathcal{H}\}$, so that a level-$h$ node, $\nu \in \mathcal{V}_h$, takes position $x_\nu \in \mathcal{D}_h$. The state variable of the root node gives the top-level executive summary of the object and lies on the lattice $\mathcal{D}_{\mathcal{H}}$. The state variables of the leaf nodes $\mathcal{V}_0$ of the graph take values on the image lattice $\mathcal{D}_0$.

The lattices are regularly spaced and the number of lattice points decreases by a factor of $q < 1$ for each level, so $|\mathcal{D}_h| = q^h|\mathcal{D}_0|$, see Figure (3)(left panel). This decrease in number of lattice points imposes the *executive summary principle*. The lattice spacing is designed so that parts do not overlap. At higher levels of the hierarchy the parts cover larger regions of the image and so the lattice spacing must be larger, and hence the number of lattice points smaller, to prevent overlapping. Note that previous work Zhu et al. (2008, 2010) was not formulated on lattices and used non-maximal suppression to achieve a similar effect.

We use the notation $\vec{x}_\nu$ to denote the state of node $\nu$ and all its descendants. This can be defined recursively by $\vec{x}_\nu = (x_\nu, x_{Ch(\nu)}, ...)$. Figure (3)(right panel) shows the object represented hierarchically where the upper level encodes the summary and the lower-levels encode the details. By a slight abuse of notation, we specify the state vector of an object $\tau_{\mathcal{H}}$ by $\vec{x}_{\tau_{\mathcal{H}}}$. *In summary*, an object $\tau_{\mathcal{H}}$ is represented by a graph with nodes $\mathcal{V}^{\tau_{\mathcal{H}}}$ and state variables $\vec{x}_{\tau_{\mathcal{H}}}$.

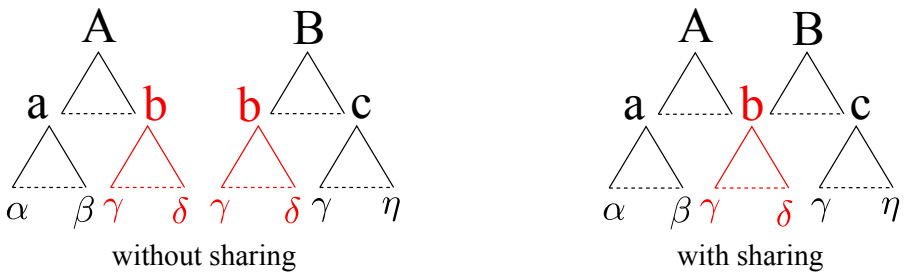



Figure 4: Part sharing. Two Level-2 parts $A$ and $B$ are composed from Lavel-1 parts $a, b$ and $c$ which, in turn, are composed from Level-0 parts $(\alpha, \beta), (\gamma, \delta)$ and $(\gamma, \zeta)$ respectively. Left Panel: the Level-2 parts are modeled separately and the fact that they share a Level-1 part is not exploited. Right Panel: In this paper we represent the part sharing explicitly and exploit it for efficient representation and inference.

### 2.4 Multiple Types of Objects, Shared Parts, and Hierarchical Dictionaries

Now suppose we have a large set of objects (each with $\mathcal{H}$ levels). We can represent each object separately by a hierarchical graph, as discussed in the last two subsections, but this is wasteful and cumbersome because it does not exploit the sharing of parts between different objects.

Instead, we represent objects and their parts by hierarchical dictionaries $\{\mathcal{M}_h : h = 0, ..., \mathcal{H}\}$. The top-level dictionary specifies the object types $\tau_{\mathcal{H}} \in \mathcal{M}_{\mathcal{H}}$. The level-$h$ dictionary specifies the types $\mathcal{M}_h$ of the parts at level $h$. The dictionary elements in $\mathcal{M}_h$ are built by compositions of $r$ elements from the dictionary $\mathcal{M}_{h-1}$ at the previous level using part-subpart composition as described in section (2.1). An object is specified by the type of its root node $\tau_{\mathcal{H}} \in \mathcal{M}_{\mathcal{H}}$. It is composed of a set $Ch(\tau_{\mathcal{H}})$ of $r$ parts from level-$\mathcal{H}-1$ with types plus spatial relations $\lambda_{\tau_{\mathcal{H}}}$ between them. Hence we express $\tau_{\mathcal{H}} = (\tau_{\mathcal{H},1}, ..., \tau_{\mathcal{H},r}; \lambda_{\tau_{\mathcal{H}}})$. In turn, these parts can be expressed in terms of lower-level subparts so that we can recursively specify the types of all parts of the object and the spatial relationships between them. More formally, any level-$h$ part $\tau_h \in \mathcal{M}_h$ can be expressed as a composition of $r$ level-$h-1$ parts $Ch(\tau_h) = \{\tau_{h,i} : i = 1, ..., r\}$, where $\tau_{h,i} \in \mathcal{M}_{h-1}$, and by spatial relations $\lambda_{\tau_h}$. For example, in Figure (1) we have $\mathcal{M}_1 = \{T, L\}$ and $\mathcal{M}_0 = \{H, V\}$, where the compositions are given by $T = (H, V, \lambda_T)$ and $L = (H, V, \lambda_L)$ (here $r = 2$).

The hierarchical dictionaries enable us to make part-sharing explicit, which will enable us to analyze its effectiveness in section (5). Part-sharing is illustrated in Figure (4) by two models $A$ and $B$ which share level-1 part $b$. Hierarchical dictionaries are shown in Figure (5) where the dictionaries are $\mathcal{M}_2 = \{A, B\}, \mathcal{M}_1 = \{a, b, c\}$ and $\mathcal{M}_0 = \{\gamma, \epsilon, \delta, \}$. For example, the level-2 part $A$ is composed from level-1 parts $a$ and $b$ (plus spatial relations, which are not shown here for simplicity), while the level-1 part $a$ is composed from level-0 parts $\delta$ and $\epsilon$ (also ignoring their spatial relations).

In summary, we represent an object $\tau_{\mathcal{H}} \in \mathcal{M}_{\mathcal{H}}$ by a graph $\mathcal{V}^{\tau_{\mathcal{H}}} = \bigcup_{h=0}^{\mathcal{H}} \mathcal{V}_h^{\tau_{\mathcal{H}}}$. There is a single node at level $\mathcal{H}$, $r$ nodes at level $\mathcal{H}-1$, and $r^{\mathcal{H}-h}$ nodes at level $h$. The state $\vec{x}_{\tau_{\mathcal{H}}}$ is specified by the state $x_{\tau_{\mathcal{H}}}$ of the root node and its descendants.

We note that the unsupervised learning algorithm in Zhu et al. (2010) automatically generates this hierarchical dictionary, as reviewed in section (6.2).

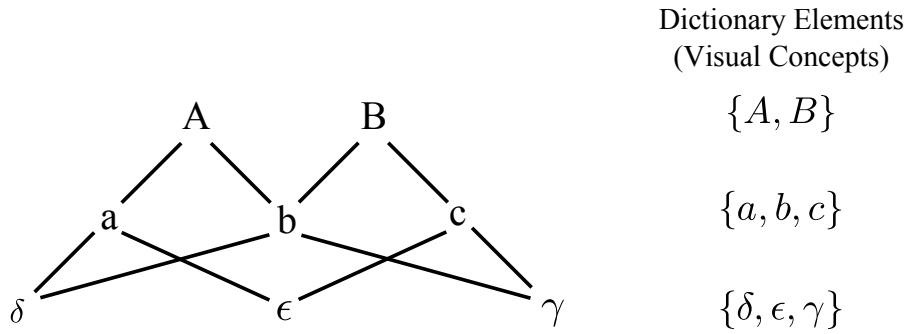



Figure 5: Representing objects in terms of hierarchical dictionaries with shared parts. There are two object models $A, B$ at the top level, three parts $a, b$ and $c$ at the middle level, and three subparts $\gamma, \delta$ and $\epsilon$ at the bottom level. The Level-2 models $A$ and $B$ are composed from Level-1 parts $a, b$ and $c$, which are composed from Level-0 parts. Here $a$ is composed from $\delta$ and $\epsilon$, $b$ is composed from $\delta$ and $\gamma$, and $c$ is composed from $\epsilon$ and $\gamma$. This illustrates part sharing, for example $\epsilon$ is shared between parts $a$ and $c$, part $b$ is shared between $A$ and $B$.

### 2.5 Probability Distributions for Objects

We specify the probability model for an object of type $\tau_{\mathcal{H}} \in \mathcal{M}_{\mathcal{H}}$ in terms of the probabilities of its part-subpart relations:

$$P(\vec{x}_{\tau_{\mathcal{H}}}|\tau_{\mathcal{H}}) = U(x_{\tau_{\mathcal{H}}}) \prod_{\nu \in \mathcal{V}^{\tau_{\mathcal{H}}}/\mathcal{V}_0^{\tau_{\mathcal{H}}}} P(x_{Ch(\nu)}|x_\nu; \tau_\nu). \tag{2}$$

Here $\mathcal{V}^{\tau_{\mathcal{H}}}/\mathcal{V}_0^{\tau_{\mathcal{H}}}$ denotes all the nodes of the graph $\mathcal{V}^{\tau_{\mathcal{H}}}$ except the leaf nodes $\mathcal{V}_0^{\tau_{\mathcal{H}}}$. $U(x_{\tau_{\mathcal{H}}})$ is the uniform distribution, which means that the object is equally likely to be anywhere in the image.

This model can be used to generate the positions of parts and subparts by sampling. This is performed by sampling the position $x_{\tau_{\mathcal{H}}}$ of the root node from $U(x_{\tau_{\mathcal{H}}})$, then sampling recursively from $P(x_{Ch(\nu)}|x_\nu; \tau_\nu)$ to get the positions of the parts and subparts until we obtain samples $\{x_\nu : \nu \in \mathcal{V}_0^{\tau_{\mathcal{H}}}\}$ for the leaf nodes.

## 3. Generative Models of Images

The models in the previous section specify distributions for the positions $\vec{x}_{\tau_{\mathcal{H}}}$ of an object $\tau_{\mathcal{H}}$, its parts and, in particular, for the positions $\{x_\nu : \nu \in \mathcal{V}_0^{\tau_{\mathcal{H}}}\}$ of the leaf nodes. In this section we describe how we can use these models to generate an image $\mathbf{I} = \{I(x) : x \in \mathcal{D}_0\}$.

This is done by introducing distributions for local image properties depending on the types and positions of the leaf nodes of the objects (e.g., the horizontal and vertical bars). We also specify a default, or background, distribution for the image properties at places where no low-level parts are present (i.e. no leaf nodes take these states).

More formally, we specify distributions $P(I(x)|\tau)$ for image property $I(x)$ at position $x \in \mathcal{D}_0$ if there is a leaf-level part $\tau \in \mathcal{M}_0$ at $x$. In addition, we specify a default *background distribution* $P(I(x)|\tau_0)$ for the image property at $x$ if no part is present, which we denote by $\tau_0$. For example,

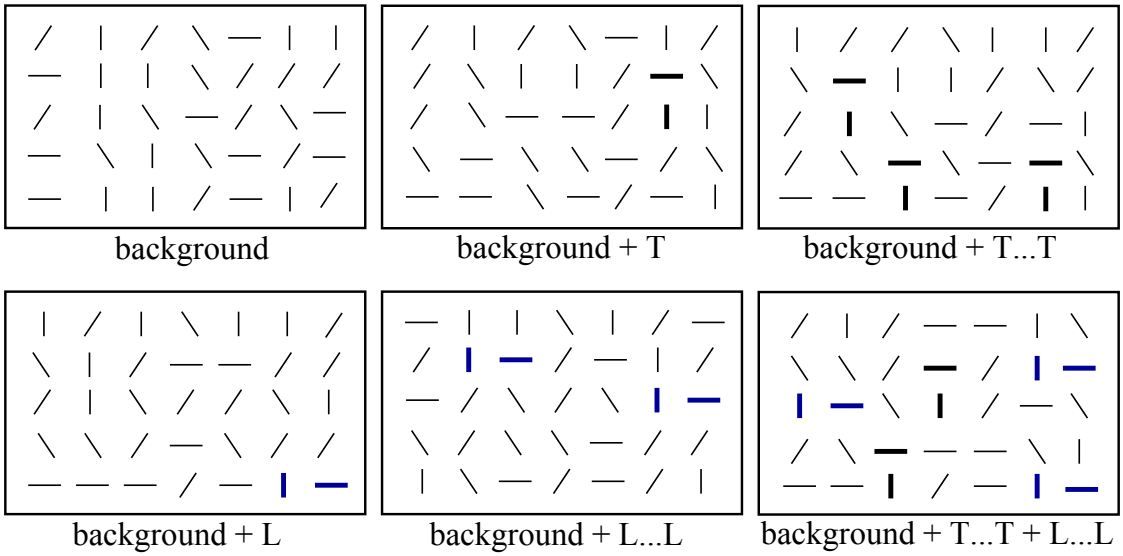



Figure 6: The generative model allows there to be several different objects in the image together with background clutter. This figure illustrates images generated by different models: (i) purely background, (ii) background with one $T$, (iii) background with several $T$'s, (iv) background with an $L$, (v) background with several $L$'s, and (vi) background with $T$'s and $L$'s.

we could use the distributions for the properties of features on, or off, edges as described in Konishi et al. (2003). Note, we only allow the leaf-level parts $\mathcal{M}_0$ to directly generate image properties.

This gives a generative model for images. We first select an object at random from $\mathcal{M}_{\mathcal{H}}$ and sample from equation (2) to get a configuration of the object and its parts including, in particular, the leaf nodes. Then for each leaf node we sample from the appropriate $P(I(x)|\tau)$ depending on the type of the leaf node. At places where no leaf node is present we sample from $P(I(x)|\tau_0)$.

Observe that we can extend this generative model in two ways. Firstly, we can sample a background image if no object is present at all, by sampling from $P(I(x)|\tau_0)$, at every position $x$. Secondly, we can sample for many objects being in the image provided they do not overlap, again by sampling from the models to find the positions of their leaf nodes and then sampling from $P(I(x)|\tau)$ or $P(I(x)|\tau_0)$ as appropriate. This can also be extended to sampling from objects and parts of objects. In this paper we mainly concentrate on the case where there is either a single object or no object in the image, although our analysis can be extended to these other more complex situations. We illustrate images sampled from different generative models in Figure (6).

### 3.1 The Generative Model for Objects and Backgrounds

We now specify the distributions more precisely. If there is no object in the image, then we generate it by sampling from the *background distribution*:

$$P_B(\mathbf{I}) = \prod_{x \in \mathcal{D}_0} P(I(x)|\tau_0). \tag{3}$$

This assumes that each image position $x$ is background and so is sampled independently from $P(I(x)|\tau_0)$.

If there is a single object $\tau_{\mathcal{H}}$ present, then the prior $P(\vec{x}_{\tau_{\mathcal{H}}}|\tau_{\mathcal{H}})$, see equation (2), gives a distribution over a set of points $\mathcal{L}(\vec{x}_{\tau_{\mathcal{H}}}) = \{x_\nu : \nu \in \mathcal{V}_0^{\tau_{\mathcal{H}}}\}$ (the leaf nodes of the graph), which are a subset of the image lattice (i.e., $x_\nu \in \mathcal{D}_0$ if $\nu \in \mathcal{V}_0^{\tau_{\mathcal{H}}}$) and specifies their types $\{\tau_\nu : \nu \in \mathcal{V}_0^{\tau_{\mathcal{H}}}\}$. We denote these leaf nodes by $\{(x, \tau(x)) : x \in \mathcal{L}(\vec{x}_{\tau_{\mathcal{H}}})\}$ where $\tau(x)$ is specified in the natural manner (i.e. if $x = x_\nu$ for $\nu \in \mathcal{V}_0^{\tau_{\mathcal{H}}}$ then $\tau(x) = \tau_\nu$). Then we sample from the distributions $P(I(x)|\tau(x))$ for the probability of the image $I(x)$ at positions $x \in \mathcal{L}(\vec{x}_{\tau_{\mathcal{H}}})$ conditioned on the type of the leaf node. We sample from the default $P(I(x)|\tau_0)$ at positions $x \in \mathcal{D}_0/\mathcal{L}(\vec{x}_{\tau_{\mathcal{H}}})$ where there is no leaf node of the object.

This specifies a likelihood function for the states $\vec{x}_{\tau_{\mathcal{H}}}$ of the object model $\tau_{\mathcal{H}}$. This likelihood function depends only on the leaf nodes $\nu \in \mathcal{V}_0^{\tau_{\mathcal{H}}}$:

$$P(\mathbf{I}|\vec{x}_{\tau_{\mathcal{H}}}) = \prod_{x \in \mathcal{L}(\vec{x}_{\tau_{\mathcal{H}}})} P(I(x)|\tau(x)) \times \prod_{x \in \mathcal{D}_0/\mathcal{L}(\vec{x}_{\tau_{\mathcal{H}}})} P(I(x)|\tau_0). \tag{4}$$

Combining this likelihood with the prior, specified in equation (2), gives the generative model for the image of an object:

$$P(\mathbf{I}|\vec{x}_{\tau_{\mathcal{H}}})P(\vec{x}_{\tau_{\mathcal{H}}}|\tau_{\mathcal{H}}). \tag{5}$$

where the prior $P(\vec{x}_{\tau_{\mathcal{H}}}|\tau_{\mathcal{H}})$ is given by equation (2).

An important quality in our analysis is the *log-likelihood ratio* of the probability of an image $\mathbf{I}$ conditioned on whether there is an object in the image of type $\tau_{\mathcal{H}}$ and configuration $\vec{x}_{\tau_{\mathcal{H}}}$ or if no object is present, $P_B(\mathbf{I})$ from equation (3). This gives a log-likelihood ratio:

$$\log \frac{P(\mathbf{I}|\vec{x}_{\tau_{\mathcal{H}}})P(\vec{x}_{\tau_{\mathcal{H}}}|\tau_{\mathcal{H}})}{P_B(\mathbf{I})} = \sum_{x \in \mathcal{L}(\vec{x}_{\tau_{\mathcal{H}}})} \log \frac{P(I(x)|\tau(x))}{P(I(x)|\tau_0)} + \sum_{\nu \in \mathcal{V}^{\tau_{\mathcal{H}}}/\mathcal{V}_0^{\tau_{\mathcal{H}}}} \log P(x_{Ch(\nu)}|x_\nu; \tau_\nu) + \log U(x_{\tau_{\mathcal{H}}}), \tag{6}$$

where the $\sum_\nu$ is performed over all part-subpart relations for object type $\tau_{\mathcal{H}}$.

All the visual tasks we study in this paper can be reduced to computing the log-likelihood ratios for different objects and different configurations of their state variables. For example, if the log-likelihood ratios are small for all object configurations then we can decide that no object is present. Alternatively, the configuration which maximizes the log-likelihood ratio determines the most probable object in the image and the most likely positions of its parts, The next section shows how we can perform all these visual tasks exactly and efficiently using a visual architecture which exploits compositionality.

## 4. The Compositional Architecture and Exact Inference

We now discuss the inference algorithms required to perform the visual tasks of determining which objects, if any, are present in the image, and estimate their positions including the positions of their subparts. These visual tasks can be decomposed into two subtasks: (i) state estimation, to determine the optimal states for each model and hence the most probable positions of the objects and their parts, and (ii) model selection, to determine whether objects are present or not and which objects are most likely. As we will show, both tasks can be reduced to calculating and comparing log-likelihood ratios which can be performed efficiently using dynamic programming methods.

Firstly we describe, in section (4.1), how to do state estimation for each object separately and then discuss how we can perform model selection to decide which objects, if any, are present in the image. Secondly, in section (4.2) we discuss how these inference tasks can be made more efficient for multiple object categories by exploiting part sharing using the hierarchical dictionaries. Thirdly, in section (4.3) we describe a parallel formulation for inference exploiting part sharing. We stress that we are performing *exact inference* and no approximations are made.

### 4.1 Inference for single objects: State Estimation and Model Selection

We first describe a standard dynamic programming algorithm for finding the optimal state of a single object model. Then we describe how the same computations can be used to perform model selection and hence applied to the detection and state estimation if the object appears multiple times in the image (provided it is non-overlapping).

Consider performing inference for a single object model $\tau_{\mathcal{H}}$ defined by equations (2) and (4). Calculating the MAP estimate of the state variables $\vec{x}_{\nu_{\mathcal{H}}}$ requires computing its most probable state $\vec{x}^*_{\tau_{\mathcal{H}}} = \arg\max_{\vec{x}_{\tau_{\mathcal{H}}}}\{\log P(\mathbf{I}|\vec{x}_{\tau_{\mathcal{H}}}) + \log P(\vec{x}_{\tau_{\mathcal{H}}}|\tau_{\mathcal{H}})\}$. This is equivalent to finding the state variables which maximize the log-likelihood ratio of $P(\mathbf{I}|\vec{x}_{\tau_{\mathcal{H}}})P(\vec{x}_{\tau_{\mathcal{H}}}|\tau_{\mathcal{H}})$ and the background distribution $P_B(\mathbf{I})$, which we specified in equation (6) (because $P_B(\mathbf{I})$ depends only on the image and is independent of the state variables). This requires estimating:

$$\vec{x}^*_{\tau_{\mathcal{H}}} = \arg\max_{\vec{x}_{\tau_{\mathcal{H}}}}\{\sum_{x\in\mathcal{L}(\vec{x}_{\tau_{\mathcal{H}}})} \log\frac{P(I(x)|\tau(x))}{P(I(x)|\tau_0)} + \sum_{\nu\in\mathcal{V}^{\tau_{\mathcal{H}}}/\mathcal{V}_0^{\tau_{\mathcal{H}}}} \log P(x_{Ch(\nu)}|x_\nu;\tau_\nu) + \log U(x_{\tau_{\mathcal{H}}})\}. \quad (7)$$

Here $\mathcal{L}$ denotes the positions of the leaf nodes of the graph, which must be determined during inference.

The optimal state $\vec{x}^*_{\tau_{\mathcal{H}}}$ can be estimated efficiently by dynamic programming. This involves a bottom-up pass which recursively computes the *local evidence* $\phi(x_\nu,\tau_\nu)$ for the *hypothesis* that there is a part $\tau_\nu$ with state variable $x_\nu$ (i.e. with executive level summary $x_\nu$). This is specified by $\phi(x_\nu,\tau_\nu) = \max_{\vec{x}_\nu/x_\nu}\{\log\frac{P(\mathbf{I}|\vec{x}_\nu)}{P_B(\mathbf{I})} + \log P(\vec{x}_\nu|\tau_\nu)\}$, where $\max_{\vec{x}_\nu/x_\nu}$ specifies that we fix the state of the position $x_\nu$ of the part $\tau_\nu$ while maximizing over the states of its descendants (recall that $\vec{x}_\nu = (x_\nu, x_{Ch(\nu)}, ...)$). The local evidence can be interpreted as the log-ratio of the hypothesis test that there is a part $\tau_\nu$ present at $x_\nu$ compared to the probability that it is not (i.e. the image properties are generated instead by the background model). The local evidences for the part hypotheses are computed bottom-up and we call it "local" because it ignores the context evidence for the part which will be provided during top-down processing (i.e. that evidence for other parts of the object, in consistent positions, will strengthen the evidence for this part). The local evidences for the part hypotheses are computed recursively by the formula:

$$\phi(x_\nu,\tau_\nu) = \max_{x_{Ch(\nu)}}\{\sum_{i=1}^{r}\phi(x_{\nu_i},\tau_{\nu_i}) + \log P(x_{Ch(\nu)}|x_\nu,\tau_\nu)\}. \quad (8)$$

At level-$\mathcal{H}$ the *bottom-up pass* outputs the *global evidence* $\phi(x_{\tau_{\mathcal{H}}},\tau_{\mathcal{H}})$ for the hypothesis that object $\tau_{\mathcal{H}}$ occurs at position $x_{\tau_{\mathcal{H}}}$. This enables us to determine the best executive summary position for the object together with the log-likelihood ratio of the best state configuration:

$$x^*_{\tau_{\mathcal{H}}} = \arg\max_{x\in\mathcal{D}_{\mathcal{H}}}\phi(x,\tau_{\mathcal{H}}), \quad \text{with evidence } \phi(x^*_{\tau_{\mathcal{H}}},\tau_{\mathcal{H}}). \quad (9)$$

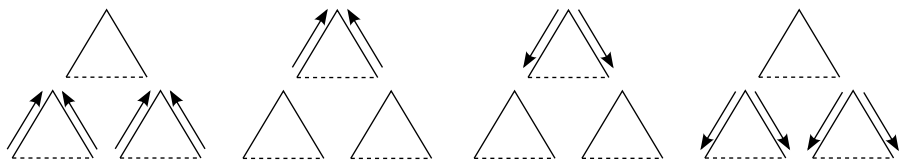


Figure 7: The feedforward (bottom-up) pass propagates hypotheses from the lowest level to the middle level (far left panel) and then from the middle level to the top level (left panel). The best top-level state is selected. The feedback (top-down) pass propagates information from the top node disambiguating the middle level nodes (right panel) and then from the middle level nodes to the lowest levels (far right panel). The bottom-up pass enables the algorithm to rapidly estimate the top-level executive summary description. The top-down pass enables high-level context to eliminate false hypotheses at the lower levels– informally, "high-level tells low-level to stop gossiping".

We can compare the log-likelihood ratio of the best state configuration to perform *model selection* either by determining whether it is above a fixed threshold $T_{\mathcal{H}}$, to determine whether the object is present or not, or to compare its value for different objects to determine which object is most likely to be in the image.

Then we can perform a *top-down pass* of dynamic programming to estimate the most probable states $\vec{x}^*_{\tau_{\mathcal{H}}}$ of the entire model by recursively performing:

$$x^*_{Ch(\nu)} = \arg\max_{x_{Ch(\nu)}} \{\sum_{i=1}^{r} \phi(x_{\nu_i}, \tau_{\nu_i}) + \log P(x_{Ch(\nu)}|x^*_\nu, \tau_\nu)\}. \tag{10}$$

This outputs the most probable configuration of this object $\tau_{\mathcal{H}}$ in the image:

$$\vec{x}^*_{\tau_{\mathcal{H}}} = x^*_{\tau_{\mathcal{H}}}, ... \tag{11}$$

It should be emphasized that the algorithm first computes the best "executive summary" description $x_{\tau_{\mathcal{H}}}$ of the object as the output of the feedforward pass, together with the log-likelihood ratio for the optimal configuration, and only later determines the optimal estimates of the lower-level states of the object in the top-down pass. Hence, the algorithm is faster at detecting that there is a cow in the right side image (estimated in the bottom-up pass) and is slower at determining the position of the feet of the cow (estimated in the top-down pass). The algorithm is also quicker at determining whether there is a cow or a horse in the image, at the end of the bottom-up pass, than it is at estimating the most likely positions of the parts of the cow or the horse. This is illustrated in Figure (7).

The algorithm also has an intuitive interpretation. The bottom up pass propagates local hypotheses for parts up the hierarchy, see Figures (7,8). These hypotheses may be extremely ambiguous at the lower levels of the hierarchy due to the local ambiguity of images. But as we ascend the hierarchy the hypotheses become increasingly unambiguous because the parts become larger and more structured. This could be analyzed by using the techniques developed in Yuille et al. (2001). From another perspective, the top-down process "binds" the parts to the subparts and hence relates to the standard binding problem of neural networks Valiant (2000).

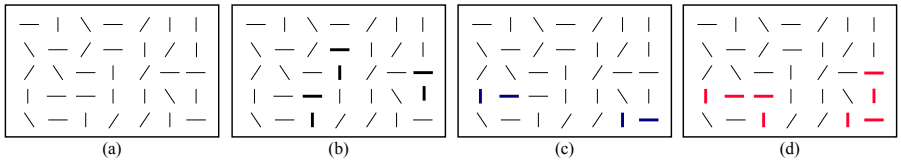



Figure 8: An alternative perspective on inference. Input image (far left panel), hypotheses for $T$'s and $L$'s (left and right panels), detections of the composite object (far right panel).

So far we have described how to use dynamic programming, with a bottom-up and top-down pass, to compute the MAP estimate of the configuration $\vec{x}_{\tau_{\mathcal{H}}}$ of an object $\tau_{\mathcal{H}}$ in the image. But this approach allows us to do more. For example, instead of detecting objects we can use the algorithm to detect parts again by using the evidence $\phi(x_\nu, \tau_\nu)$ as a log-likelihood ratio test to determine if the part is present or not. The log-likelihood test will tend to have a large error rate for low level parts (because of their ambiguity) but will increasingly have fewer errors for larger parts since these are less ambiguous. Intuitively, it is possible for a human to confuse the leg of a cow with the leg of a horse but it is extremely unlikely that an entire horse and cow cannot be distinguished. Using this strategy it may be possible to decide that an object part is present or not without waiting for the context provided by the top-down processing. Making premature decisions like this may cause problems, however, which we analyze in section (6.3).

In addition, we can compute the probability that the object occurs several times in the image, by computing the set $\{x_{\tau_{\mathcal{H}}} : \phi(x^*_{\tau_{\mathcal{H}}}, \tau_{\mathcal{H}}) > T_{\mathcal{H}}\}$, to compute the "executive summary" descriptions for each object (e.g., the coarse positions of each object). We then perform the top-down pass initialized at each coarse position (i.e. at each point of the set described above) to determine the optimal configuration for the states of the objects. Hence, we can reuse the computations required to detect a single object in order to detect multiple instances of the object (provided there are no overlaps). The number of objects in the image is determined by the log-likelihood ratio test with respect to the background model. It can be shown that this is equivalent to performing optimal inference simultaneously over a set of different generative models of the image, where one model assumes that there is one instance of the object in the image, another model assumes there are two (non-overlapping), and so on.

### 4.2 Compositional Architecture: Inference on Multiple Objects by Part Sharing using the Hierarchical Dictionaries

The previous section performed MAP estimation, and computed log-likelihood ratios, for each object separately. But this is wasteful since if the objects share parts because it computes the same quantities multiple times. This section describes how we can exploit part sharing to perform these computations more efficiently for many objects simultaneously.

The main idea is that we only need to compute the local evidence $\phi(x_\nu, \tau_\nu)$ for each part $\tau_\nu \in \mathcal{M}_h$ *once* independently of how many objects the part occurs in (or how many times it occurs within each object). This means that we can compute the evidence for all objects at all positions $\{\phi(\vec{x}_{\tau_{\mathcal{H}}}, \tau_{\mathcal{H}}) : \vec{x}_{\tau_{\mathcal{H}}} \in \mathcal{D}_{\mathcal{H}},\ \tau_{\mathcal{H}} \in \mathcal{M}_{\mathcal{H}}\}$ by recursively computing the evidences $\{\phi(x_\nu, \tau_\nu) : x_\nu \in \mathcal{D}_h,\ \tau_\nu \in \mathcal{M}_h\}$ using equation (8) for all $x_\nu \in \mathcal{D}_h,\ \tau_\nu \in \mathcal{M}_h$.

We then perform model selection at the top-level by thresholding the evidence for each object at each position. More precisely, for each object $\tau_{\mathcal{H}} \in \mathcal{M}_{\mathcal{H}}$ we determine the set of positions $\mathcal{S}_{\tau_{\mathcal{H}}} = \{x_{\tau_{\mathcal{H}}} \in \mathcal{D}_{\mathcal{H}} : s.t.\ \phi(x_{\tau_{\mathcal{H}}}, \tau_{\mathcal{H}}) > T_{\mathcal{H}}\}$ where the evidence for that object is above threshold. Then we perform the top-down pass, equation (10), starting at all positions $\mathcal{S}_{\tau_{\mathcal{H}}}$ for all $\tau_{\mathcal{H}} \in \mathcal{M}_{\mathcal{H}}$. In short, we perform bottom-up inference to detect which objects are present and the positions of the executive summaries. Then we perform top-down processing to estimate the positions of the parts, the full configurations, of the detected objects. Note, if this process detects two or more objects in the same position then we will need to compare their evidences to select which one is most likely, but our framework assumes that this is unlikely to occur because there is little ambiguity for complete objects.

Note that we are performing exact inference over multiple object models at the same time. This may be un-intuitive to some readers because this corresponds to doing exact inference over a probability model which can be expressed as a graph with a large number of closed loops, see Figure (4). We note that this applies only to a subclass of compositional models including Zhu et al. (2008, 2010). The main point is that part-sharing enables us perform inference efficiently for many models at the same time.

### 4.3 Parallel Formulation of the Compositional Architecture

Finally, we observe that all the computations required for performing inference on multiple objects can be parallelized.

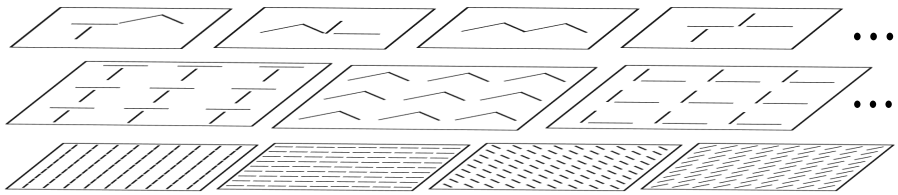


Figure 9: Parallel Hierarchical Implementation. Bottom Row: Four Level-0 parts are placed at each point in the image lattice $\mathcal{D}_0$. Middle Row: Level-1 parts are placed at each point on the coarser lattice $\mathcal{D}_1$ (we show three Level-1 models). Top Row: object models are placed at the nodes of the top level lattice ($\mathcal{H} = 2$ in this figure). This can be thought of as non-linear receptive fields analogous to bio-inspired hierarchical models.

Parallelization is possible for both the bottom-up and the top-down passes. Firstly, the bottom-up pass requires calculating the $\phi(x_\nu, \tau_\nu)$'s at each layer-$h$ using equation (8). These calculations are done separately for each type $\tau_\nu \in \mathcal{M}_h$ and at each spatial position $x_\nu \in \mathcal{D}_h$, see equation (8). They depend only on input from local subregions of layer-$(h-1)$. Hence they can be computed in parallel. This enables us to compute the global evidence for objects at the top level: $\phi(x_{\tau_{\mathcal{H}}}, \tau_{\mathcal{H}})$ for all $x_{\tau_{\mathcal{H}}} \in \mathcal{D}_{\mathcal{H}}$ and $\tau_{\mathcal{H}} \in \mathcal{M}_{\mathcal{H}}$. We threshold at each node $x_{\tau_{\mathcal{H}}} \in \mathcal{D}_{\mathcal{H}}$ to determine if the object is detected, which is also a local operation. Secondly, the top-down pass is also parallelizable because its operation, see equation (10), is also local. Note that top-down processing is only done starting from nodes at level-$\mathcal{H}$ where objects have been detected.

This leads to a graphical architecture which contains $|\mathcal{M}_h| \times |\mathcal{D}_h|$ nodes at level $h$, and hence a total of $\sum_{h=0}^{\mathcal{H}} |\mathcal{M}_h| \times |\mathcal{D}_h|$ nodes. There are a total of $r \sum_{h=1}^{\mathcal{H}} |\mathcal{M}_h| \times |\mathcal{D}_h|$ connections. These

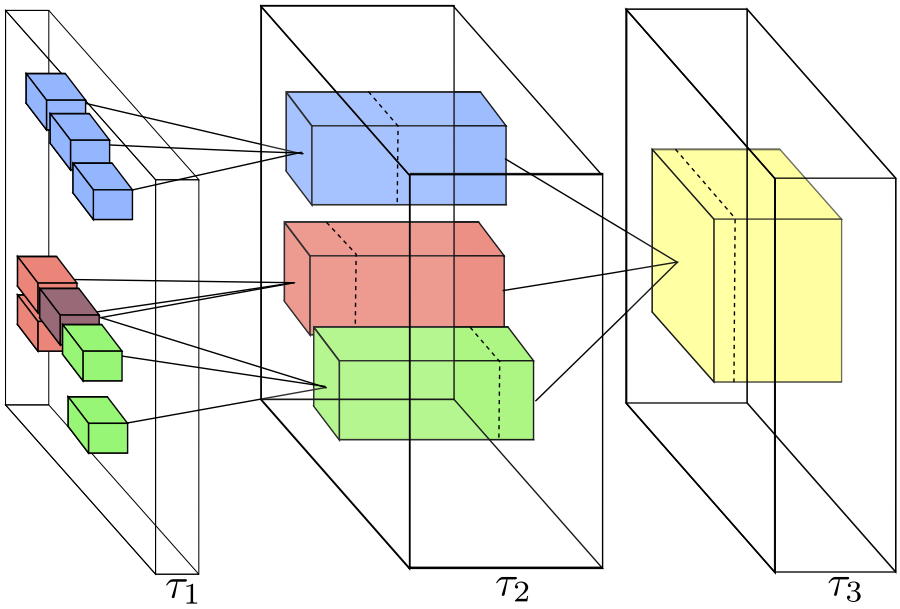



Figure 10: This figure illustrates the parallel hierarchical architecture. At each level there are nodes tuned to different types of parts (similar to the different features occurring in bio-inspired models).

connections are local and "convolutional" in the sense that they are the same at different places in the image. Nodes at level-$h$ can be represented by $(\tau_\nu, x_\nu)$ and so they represent a part of type $\tau_\nu \in \mathcal{M}_h$ at position $x_\nu \in \mathcal{D}_h$. This is illustrated in Figures (9) and (10).

We can think of the nodes at different levels of the hierarchy as being non-linear "receptive fields". At level-0, these receptive fields are tuned to specific image properties (e.g., horizontal or vertical bars). Note that the receptive fields are highly non-linear (i.e. they do not obey the superposition principle). Nevertheless they are broadly speaking, tuned to image stimuli which have the mean shape of the corresponding part $\tau_\nu$. In agreement, with findings about mammalian cortex, the receptive fields become more sensitive to image structure (e.g., from bars, to more complex shapes) at increasing levels. Moreover, their sensitivity to spatial position decreases because at higher levels the models only encode the executive summary descriptions, on coarser lattices, while the finer details of the object are represented more precisely at the lower levels. Moreover, they are influenced both by bottom-up processing (during the bottom-up pass) and by top-down processing (during the top-down pass). The bottom-up processing computes the local evidence while the top-down pass modifies it by the high-level context.

The computations required by this parallel implementation are illustrated in Figure (11). Each step of the bottom-up pass is performed by a two-layer network, see Figure (11), where the first layer performs an AND operation (to compute the local evidence for a specific configuration of the child nodes) and the second layer performs an OR, or $\max$ operation, to determine the local evidence (by max-ing over the possible child configurations). The top-down pass only has to perform an $\arg\max$ computation to determine which child configuration gave the best local evidence.

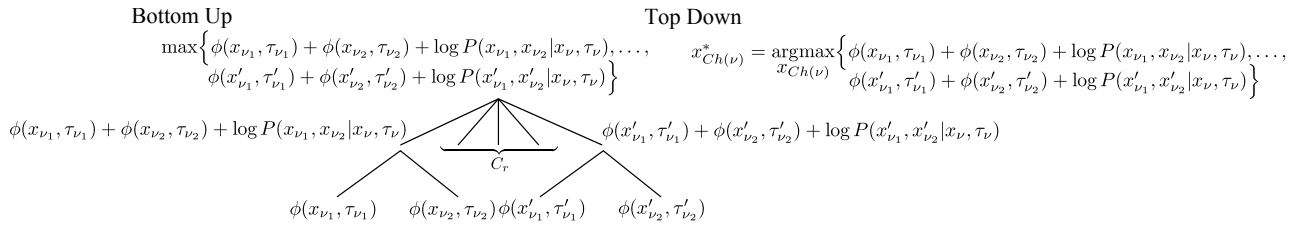



Figure 11: Parallel implementation of Dynamic Programming. The left part of the figure shows the bottom-up pass of dynamic programming. The local evidence for the parent node is obtained by taking the maximum of the scores of the $C_r$ possible states of the child nodes. This can be computed by a two-layer network where the first level computes the scores for all $C_r$ child node states, which can be done in parallel, and the second level compute the maximum score. This is like an AND operation followed by an OR. The top-down pass requires the parent node to select which of the $C_r$ child configurations gave the maximum score, and suppressing the other configurations.

## 5. Complexity Analysis

We now analyze the complexity of the inference algorithms for performing the tasks of object detection and object parsing. Firstly, we analyze complexity for a single object (without part-sharing). Secondly, we study the complexity for multiple objects with shared parts. Thirdly, we consider the complexity of the parallel implementation.

The complexity is expressed in terms of the following quantities: (I) The size $|\mathcal{D}_0|$ of the image. (II) The scale decrease factor $q$ (enabling executive summary). (III) The number $\mathcal{H}$ of levels of the hierarchy. (IV) The sizes $\{|\mathcal{M}_h| : h = 0, ..., \mathcal{H}\}$ of the hierarchical dictionaries. (V) The number $r$ of subparts of each part. (VI) The number $C_r$ of possible part-subpart configurations.

### 5.1 Complexity for Single Objects and Ignoring Part Sharing

This section estimates the complexity of inference $N_{s_o}$ for a single object and the complexity $N_{m_o}$ for multiple objects when part sharing is not used. These results are for comparison to the complexities derived in the following section which use part sharing.

The inference complexity for a single object requires computing: (i) the number $N_{bu}$ of computations required by the bottom-up pass, (ii) the number $N_{m_s}$ of computations required by model selection at the top-level of the hierarchy, and (iii) the number $N_{td}$ of computations required by the top-down pass.

The complexity $N_{bu}$ of the bottom-up pass can be computed from equation (8). This requires a total of $C_r$ computations for each position $x_\nu$ for each level-$h$ node. There are $r^{\mathcal{H}-h}$ nodes at level-$h$ and each can take $|\mathcal{D}_0|q^h$ positions. This gives a total of $|\mathcal{D}_0|C_r q^h r^{\mathcal{H}-h}$ computations at level $h$. This can be summed over all levels to yield:

$$N_{bu} = \sum_{h=1}^{\mathcal{H}} |\mathcal{D}_0| C_r r^{\mathcal{H}} (q/r)^h = |\mathcal{D}_0| C_r r^{\mathcal{H}} \sum_{h=1}^{\mathcal{H}} (q/r)^h = |\mathcal{D}_0| C_r \frac{q r^{\mathcal{H}-1}}{1-q/r}\{1-(q/r)^{\mathcal{H}}\}. \quad (12)$$

Observe that the main contributions to $N_{bu}$ come from the first few levels of the hierarchy because the factors $(q/r)^h$ decrease rapidly with $h$. This calculation uses $\sum_{h=1}^{\mathcal{H}} x^h = \frac{x(1-x^{\mathcal{H}})}{1-x}$. For large $\mathcal{H}$ we can approximate $N_{bu}$ by $|\mathcal{D}_0|C_r \frac{qr^{\mathcal{H}-1}}{1-q/r}$ (because $(q/r)^{\mathcal{H}}$ will be small).

We calculate $N_{m_s} = q^{\mathcal{H}}|\mathcal{D}_0|$ for the complexity of model selection (which only requires thresholding at every point on the top-level lattice).

The complexity $N_{td}$ of the top-down pass is computed from equation (10). At each level there are $r^{\mathcal{H}-h}$ nodes and we must compute $C_r$ computations for each. This yields complexity of $\sum_{h=1}^{\mathcal{H}} C_r r^{\mathcal{H}-h}$ for each possible root node. There are at most $q^{\mathcal{H}}|\mathcal{D}_0|$ possible root nodes (depending on the results of the model selection stage). This yields an upper bound:

$$N_{td} \leq |\mathcal{D}_0|C_r q^{\mathcal{H}} \frac{r^{\mathcal{H}-1}}{1-1/r}\{1 - \frac{1}{r^{\mathcal{H}-1}}\}. \tag{13}$$

Clearly the complexity is dominated by the complexity $N_{bu}$ of the bottom-up pass. For simplicity, we will bound/approximate this by:

$$N_{s_o} = |\mathcal{D}_0|C_r \frac{qr^{\mathcal{H}-1}}{1-q/r}. \tag{14}$$

Now suppose we perform inference for multiple objects simultaneously without exploiting shared parts. In this case the complexity will scale linearly with the number $|\mathcal{M}_{\mathcal{H}}|$ of objects. This gives us complexity:

$$N_{m_o} = |\mathcal{M}_{\mathcal{H}}||\mathcal{D}_0|C_r \frac{qr^{\mathcal{H}-1}}{1-q/r}. \tag{15}$$

### 5.2 Computation with Shared Parts in Series and in Parallel

This section computes the complexity using part sharing. We first study complexity for the standard serial implementation of part sharing and then for the parallel implementation.

Now suppose we perform inference on many objects with part sharing using a serial computer. This requires performing computations over the part-subpart compositions between elements of the dictionaries. At level $h$ there are $|\mathcal{M}_h|$ dictionary elements. Each can take $|\mathcal{D}_h| = q^h|\mathcal{D}_0|$ possible states. The bottom-up pass requires performing $C_r$ computations for each of them. This gives a total of $\sum_{h=1}^{\mathcal{H}} |\mathcal{M}_h|C_r|\mathcal{D}_0|q^h = |\mathcal{D}_0|C_r \sum_{h=1}^{\mathcal{H}} |\mathcal{M}_h|q^h$ computations for the bottom-up process. The complexity of model selection is $|\mathcal{D}_0|q^{\mathcal{H}} \times (|\mathcal{M}_{\mathcal{H}}| + 1)$ (this is between all the objects, and the background model, at all points on the top lattice). As in the previous section, the complexity of the top-down process is less than the complexity of the bottom-up process. Hence the complexity for multiple objects using part sharing is given by:

$$N_{p_s} = |\mathcal{D}_0|C_r \sum_{h=1}^{\mathcal{H}} |\mathcal{M}_h|q^h. \tag{16}$$

Next consider the parallel implementation. In this case almost all of the computations are performed in parallel and so the complexity is now expressed in terms of the number of "neurons" required to encode the dictionaries, see Figure (9). This is specified by the total number of dictionary elements multiplied by the number of spatial copies of them:

$$N_n = \sum_{h=1}^{\mathcal{H}} |\mathcal{M}_h|q^h|\mathcal{D}_0|. \tag{17}$$

The computation, both the forward and backward passes of dynamic programming, are linear in the number $\mathcal{H}$ of levels. We only need to perform the computations illustrated in Figure (11) between all adjacent levels.

Hence the parallel implementation gives speed which is linear in $\mathcal{H}$ at the cost of a possibly large number $N_n$ of "neurons" and connections between them.

### 5.3 Advantages of Part Sharing in Different Regimes

The advantages of part-sharing depend on how the number of parts $|\mathcal{M}_h|$ scales with the level $h$ of the hierarchy. In this section we consider three different regimes: (I) The *exponential growth regime* where the size of the dictionaries increases exponentially with the level $h$. (II) The *empirical growth regime* where we use the size of the dictionaries found experimentally by compositional learning Zhu et al. (2010). (III) The *exponential decrease regime* where the size of the dictionaries decreases exponentially with level $h$. For all these regimes we compare the advantages of the serial and parallel implementations using part sharing by comparison to the complexity results obtained without sharing.

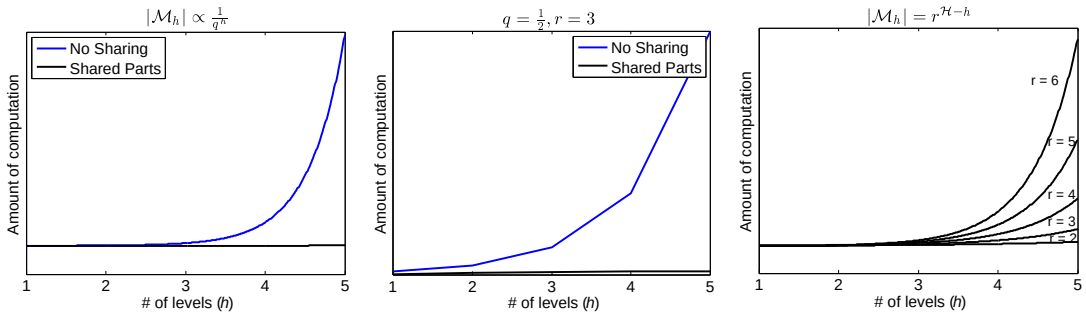
Exponential growth of dictionaries is a natural regime to consider. It occurs when subparts are allowed to combine with all other subparts (or a large fraction of them) which means that the number of part-subpart compositions is polynomial in the number of subparts. This gives exponential growth in the size of the dictionaries if it occurs at different levels (e.g., consider the enormous number of objects that can be built using lego).

An interesting special case of the exponential growth regime is when $|\mathcal{M}_h|$ scales like $1/q^h$ (recall $q < 1$), see Figure (12) (left panel). In this case the complexity of computation for serial part-sharing, and the number of neurons required for parallel implementation, scales only with the number of levels $\mathcal{H}$. This follows from equations (16) and (17). But nevertheless the number of objects that can be detected scales exponentially with $\mathcal{H}$, as $(1/q)^{\mathcal{H}}$. By contrast, the complexity of inference without part-sharing also scales exponentially as $(1/q)^{\mathcal{H}}$, see equation (15), because we have to perform a fixed number of computations, given by equation (14), for each of an exponential number of objects. This is summarized by the following result.

*Result 1*: If the number of shared parts scales exponentially by $|\mathcal{M}_h| \propto \frac{1}{q^h}$ then we can perform inference for order $(1/q)^{\mathcal{H}}$ objects using part sharing in time linear in $\mathcal{H}$, or with a number of neurons linear in $\mathcal{H}$ for parallel implementation. By contrast, inference without part-sharing requires exponential complexity.

To what extent is exponential growth a reasonable assumption for real world objects? This motivates us to study the empirical growth regime using the dictionaries obtained by the compositional learning experiments reported in Zhu et al. (2010). In these experiments, the size of the dictionaries increased rapidly at the lower levels (i.e. small $h$) and then decreased at higher levels (roughly consistent with the findings of psychophysical studies – Biederman, personal communication). For these "empirical dictionaries" we plot the growth, and the number of computations at each level of the hierarchy, in Figure (12)(center panel). This shows complexity which roughly agrees with the exponential growth model. This can be summarized by the following result:

*Result 2*: If $|\mathcal{M}_h|$ grows slower than $1/q^h$ and if $|\mathcal{M}_h| < r^{\mathcal{H}-h}$ then there are gains due to part sharing using serial and parallel computers. This is illustrated in Figure (12)(center panel) based on the dictionaries found by unsupervised computational learning Zhu et al. (2010). In parallel implementations, computation is linear in $\mathcal{H}$ while requiring a limited number of nodes ("neurons").




(a) (b) (c)

Figure 12: The curves are plotted as a function of $h$. Left panel: The first plot is the case where $M_h = a/(q^h)$. So we have a constant cost for the computations, when we have shared parts. Center panel: This plot is based on the experiment of Zhu et al. (2010). Right panel: The third plot is the case where $M_h$ decreases exponentially. The amount of computation is the same for the shared and non-shared cases.

Finally we consider the exponential decrease regime. To motivate this regime, suppose that the dictionaries are used to model image appearance, by contrast to the dictionaries based on geometrical features such as bars and oriented edges (as used in Zhu et al. (2010)). It is reasonable to assume that there are a large number of low-level dictionaries used to model the enormous variety of local intensity patterns. The number of higher-level dictionaries can decrease because they can be used to capture a cruder summary description of a larger image region, which is another instance of the executive summary principle. For example, the low-level dictionaries could be used to provide detailed modeling of the local appearance of a cat, or some other animal, while the higher-level dictionaries could give simpler descriptions like "cat-fur" or "dog-fur" or simply "fur". In this case, it is plausible that the size of the dictionaries decreases exponentially with the level $h$. The results for this case emphasize the advantages of parallel computing.

*Result 3*: If $|\mathcal{M}_h| = r^{\mathcal{H}-h}$ then there is no gain for part sharing if serial computers are used, see Figure (12)(right panel). Parallel implementations can do inference in time which is linear in $\mathcal{H}$ but require an exponential number of nodes ("neurons").

Result 3 may appear negative at first glance even for the parallel version since it requires an exponentially large number of neurons to encode the lower level dictionaries. But it may relate to one of the more surprising facts about the visual cortex in monkeys and humans, if we identify the nodes of the compositional models with neurons in the visual cortex, namely that the first two visual areas, V1 and V2, where low-level dictionaries would be implemented are enormous compared to the higher levels such as IT where object detection takes places. Current models of V1 and V2 mostly relegate them to being a large filter bank which seems paradoxical considering their size. For example, Lennie (1998) has stated when reviewing the functions of V1 and V2 "perhaps the most troublesome objection to the picture I have delivered is that an enormous amount of cortex is used to achieve remarkably little". Perhaps the size of these visual areas is because they are encoding dictionaries of visual appearance.

## 6. Discussion

This section discusses three important topics. Section (6.1) discusses the relation of compositional models to the more traditional bio-inspired hierarchical models, such as deep convolutional networks. In section (6.2) we briefly describe how compositional models can be learnt in an unsupervised manner. Section (6.3) describes how to extend compositional models to make them more robust, and describes an analysis of robustness which is presented in detail in Appendix A.

### 6.1 Compositional Models and Bio-inspired Hierarchical Architectures

The parallel compositional architecture described in the last two sections has several similarities to bio-inspired models and in particular to deep convolutional networks Krizhevsky et al. (2012). The graphical structures of the models are similar and the nodes at each level of the hierarchy are indexed by position in the image lattice and by the type of the part, or correspondingly, the index of the feature detector (i.e. the class of feature detectors in a convolutional network corresponds to the set of part types for compositional models).

But there are also several differences which we now discuss. Compositional models were developed from the literature on deformable template models of objects where the representation of objects parts and spatial relations is fundamental. These deformable template models are designed to detect the parts of objects and not simply to detect which objects are present in an image. Compositional models, and the closely related grammar models, enable parts to be shared between multiple objects while, by contrast, bio-inspired models share features.

In particular, compositional models have explicit part-subpart relationships which enable them to represent spatial relations explicitly and to perform tasks like parsing in addition to detection. These part-subpart relations are learnt by hierarchical clustering algorithms Zhu et al. (2008, 2010) so that a part is composed from $r$ subparts and the spatial relations between the subparts are explicitly modeled. This differs from the way that features are related in convolutional networks. The difference is most apparent at the final layers where convolutional networks are fully connected (i.e. all features at the top-level can be used for all objects) while compositional models still restrict the objects to be composed of $r$ subparts. In other words, the final level of a compositional model is exactly the same as all the previous levels (i.e. it does not extract hierarchical features using one learning algorithm and then learn a classifier on top).

In addition, the compositional models are generative. This has several advantages including the possibility of dealing robustly with occlusion and missing parts, as we discuss in the next section, and the ability (in principle) to synthesize images of the object by activating top-level object nodes. These abilities suggest models for visual attention and other top-down processing.

Using this compositional architecture, the inference is performed by a bottom-up process which propagates hypotheses (for the states of nodes at the low-levels of the hierarchy) up the hierarchy. As they ascend the hierarchy they have access to more information about the image (i.e. the higher level nodes represent larger parts of the image) and hence become less ambiguous. A top-down process is used to disambiguate the lower level hypotheses using the context provided by the higher level nodes – "high-level tells low-level to stop gossiping". We stress that this is exact inference – similar (and sometimes identical) to dynamic programming on probabilistic context free grammars. Hence, as implemented in the compositional architecture, we see that we can have generative models of many objects but we can nevertheless perform inference rapidly by bottom-up processing followed

by top-down stage which determines the positions of the object parts/subparts by eliminating false hypotheses.

More technically, we now specify a rough translation between compositional models and deep convolutional neural networks (DCNNs). The types $\tau$ in compositional models translate into the feature channels of DCNNs. The local evidence $\phi(x_\nu, \tau_\nu)$ for type $\tau$ at position $x_\nu$ corresponds to the activation $z_{i,k}$ of the $k^{th}$ channel at position $i$ in a DCNN. If we ignore max-pooling, for simplicity, the DCNN will have update rule:

$$\phi(x_\nu, \tau_\nu) = \sum_{i, x_{\nu_i}} \omega_{i, x_{\nu_i}} \max\{0, \phi(x_{\nu_i}, \tau_{\nu_i})\},$$

with weights $\omega_{i,x_{\nu_i}}$ and where the summation is over the child nodes $\nu_i$ and over their positions $x_{\nu_i}$. The max-pooling replaces $\phi(x_{\nu_i}, \tau_{\nu_i})$ by $\max_{y \in Nbh(x_{\nu_i})} \phi(y, \tau_{\nu_i})$, where $Nbh(x_{\nu_i})$ is a spatial neighbourhood centered on $x_{\nu_i}$. Hence both update rules involve maximization steps but for DCNNs they are independent for each feature channel and for compositional models they depend on the states of all the children. DCNN includes weights $\omega_{i,x_{\nu_i}}$, which are trained discriminatively by backpropagation, while compositional models use probabilities $P(x_{Ch(\nu)}|x_\nu, \tau_\nu)$. The learning for compositional models is unsupervised and is discussed in section (6.2).

### 6.2 Unsupervised Learning

This section briefly sketches the unsupervised learning algorithm. We refer to Zhu et al. (2008, 2010); Yuille (2011) for more details. The strategy is hierarchical clustering which proceeds by recursively learning the dictionaries.

We start with a level-0 dictionary of parts $\tau \in \mathcal{M}_0$ with associated models $P(I(x)|\tau)$ and a default distribution $P(I(x)|\tau_0)$. These are assumed as inputs to the learning process. But they could be learnt by unsupervised learning as in Papandreou et al. (2014); Mao et al. (2014).

Then we select a threshold $T_0$ and for each level-0 type $\tau \in \mathcal{M}_0$ we detect a set of image positions $\{x_i^\tau : i = 1, ..., N\}$ where the log-likelihood test for $\tau$ is above threshold, i.e. $\log \frac{P(I(x_i^\tau)|\tau)}{P(I(x)|\tau_0)} > T_0$. This corresponds to the set of places where the part has been detected. This process depends on the threshold $T_0$ which will be discussed later.

Next we seek compositions of parts that occur frequently together with regular spatial arrangements. This is performed by clustering the detected parts using a set of probabilistic models of form $h(x_{\tau_1}, ..., x_{\tau_r}; \lambda)$, see equation (1), to estimate the $\lambda$'s (the clustering is done separately for different combinations of level-0 types). This yields the level-1 dictionary $\mathcal{M}_1$, where each level-1 part is described by a probability distribution specified by the types of its children $\tau_1, ..., \tau_r$ and the spatial relations given by $h(., ., ...; \lambda)$. We then repeat the process. More specifically, we detect level-1 parts using the log-likelihood ratio test with a threshold $T_1$ and perform clustering to create the level-2 dictionary $\mathcal{M}_2$. We continue recursively using detected parts as inputs to another spatial clustering stage. The process terminates automatically when we fail to detect any new clusters. This will depend on our choice of clusters, the detection thresholds, and a threshold on the number of instances needed to constitute a cluster. For more details see Zhu et al. (2008, 2010). The whole process can be interpreted as a breadth first search through the space of possible generative models of the image, see Yuille (2011).

### 6.3 Robust Compositional Models

It is also important to study the robustness of compositional models when some parts are undetected. This is an important issue because objects can often be partially occluded in which case some parts are not observed directly. Also some forms of inference require thresholding the log-likelihood ratios of objects to determine if they have been detected (i.e. without waiting to detect the complete object) and we need to understand what happens for the false negatives when we threshold the log-likelihoods. As described above, thresholding is also used during learning, which gives more motivation for understanding the errors it causes and how to minimize them. Finally, we note that any realistic neural implementation of compositional models must be robust to failures of neural elements.

These issues were addressed in previous work Zhu et al. (2008, 2010) which showed empirically that compositional models could be made robust to some errors of this type. This section briefly describes some of these extensions. In addition to the motivations given above, we also want ways to extend compositional models to allow for parts to have variable numbers of subparts or to be partially invariant to image transformations. Firstly, the part-subpart distributions $P(\vec{x}_{Ch(\nu)}|x_\nu, \tau_\nu)$ were made insensitive to rotations and expansions in the image plane. Secondly, the 2-out-of-3 rule (described below) made the model robust to failure to detect parts or to malfunctioning neurons. Thirdly, the imaging terms could be extended so that the model generates image features (instead of the image values directly) and could include direct image input to higher levels. All these extensions were successfully tested on natural images Zhu et al. (2008, 2010).

The 2-out-of-3 rule is described as follows. In our implementations Zhu et al. (2008, 2010) a part-subpart composition has three subparts. The 2-out-of-3 rule allowed the parent node to be activated if only two subparts were detected, even if the image response at the third part is inconsistent with the object (e.g., if the object is partially occluded). The intuition is that if two parts are detected then the spatial relations between the parts, embedded in the term $h(\vec{x}; \lambda_\nu)$ of $P(\vec{x}_{Ch(\nu)}|x_\nu, \tau_\nu)$, is sufficient to predict the position of the third part. This can be used during inference while paying a penalty for not detecting the part. From another perspective, this is equivalent to having a mixture of different part-subparts compositions where the part could correspond to three subparts or two subparts. This can be extended to allow a larger range of possible subpart combinations thereby increasing the flexibility. Note that this would not affect the computational time for a parallel model.

We analyzed the robustness of the 2-out-of-3 rule when parts are missing. This analysis is presented in appendix A. It shows that provided the probability of detecting each part is above a threshold value (assuming the part is present) then the probability of detecting the entire object correctly tends to 1 as the number of levels of the hierarchy increases.

## 7. Summary

This paper provides a complexity analysis of what is arguably one of the most fundamental problem of visions – how, a biological or artificial vision system could rapidly detect and recognize an enormous number of different objects. We focus on a class of hierarchical compositional models Zhu et al. (2008, 2010) whose formulation makes it possible to perform this analysis. We conjecture that similar results, exploiting the sharing of parts and hierarchical distributed representations, will apply to the more sophisticated models needed to address the full complexity of object detection and parsing. Similar results may apply to related bio-inspired hierarchical models of vision (e.g., those cited in the introduction). We argue that understanding complexity is a key issue in the design of

artificial intelligent systems and understanding the biological nature of intelligence. Indeed perhaps the major scientific advances this century will involve the study of complexity Hawking (2000). We note that complexity results have been obtained for other visual tasks Blanchard and Geman (2005),Tsotsos (2011).

Technically this paper has required us to re-formulate compositional models to define a compositional architecture which facilitates the sharing of parts. It is noteworthy that we can effectively perform exact inference on a large number of generative image models (of this class) simultaneously using a single architecture. It is interesting to see if this ability can be extended to other classes of graphical models and be related to the literature on rapid ways to do exact inference, see Chavira et al. (2004).

Finally, we note that the parallel inference algorithms used by this class of compositional models have an interesting interpretation in terms of the bottom-up versus top-down debate concerning processing in the visual cortex DiCarlo et al. (2012). The algorithms have rapid parallel inference, in time which is linear in the number of layers, and which rapidly estimates a coarse "executive summary" interpretation of the image. The full interpretation of the image takes longer and requires a top-down pass where the high-level context is able to resolve ambiguities which occur at the lower levels. Of course, for some simple images the local evidence for the low level parts is sufficient to detect the parts in the bottom-up pass and so the top-down pass is not needed. But more generally, in the bottom-up pass the neurons are very active and represent a large number of possible hypotheses which are pruned out during the top-down pass using context, when "high-level tells low-level to stop gossiping".

## Appendix A

We analyze the 2-out-of-3 rule to show its robustness to missing parts. Consider two layers of part-subpart compositions with a parent part, three subparts, and nine sub-subparts. The 2-out-of-3 rule enables us to detect the part even in cases where only four of the nine sub-subparts are detected (provided that one subpart has no sub-subparts detected and the remaining two subparts have two sub-subparts detected each). To study this in more detail, let $\rho$ be the probability of detecting a subpart and $f(\rho)$ be the probability of detecting the part. Using the 2-out-of-3 rule, we obtain the update rule:

$$\rho_{h+1} = f(\rho_h), \text{ with } f(\rho) = \rho^3 + 3\rho^2(1-\rho). \tag{18}$$

We can analyze the robustness of the rule by studying the behavior of the iterative map $\rho_{t+1} = f(\rho_t)$. It can be calculated that there are fixed points at $\rho = 0, 0.5, 1$. Observe that $f(\rho) > \rho$ for $0.5 < \rho < 1$ and $f(\rho) < \rho$ for $0 < \rho < 0.5$, see Figure (13). Hence if the initial value of $\rho < 0.5$ (i.e. the detectors at the leaf nodes miss the object more than half the time) then the iterative map converges to zero and we cannot detect the object. Conversely, if $\rho > 0.5$ (the initial detectors miss parts less than half the time) then the iterative map converges to 1 and we always detect the object if there are a sufficient number of levels.

We performed simulations to verify this result and to estimate how many levels are needed to obtain almost perfect detection as a function of $\rho$. These are shown in Figure (13)(right panel). Observe that if $\rho > 0.6$ then only a small number of levels are needed to get almost perfect performance.

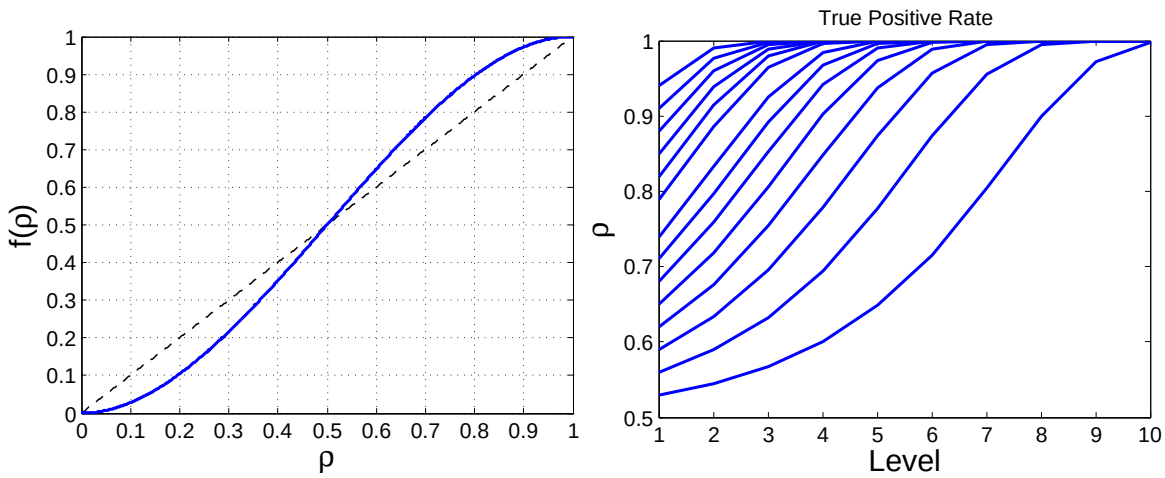



Figure 13: Left Panel: we plot $f(\rho)$ as a function of $\rho$, showing fixed points at $\rho = 0, 0.5, 1$, and that $f(\rho) > \rho$ for $0.5 < \rho < 1$, but $f(\rho) < \rho$ for $0 < \rho < 0.5$. Right Panel: we show the true positive rate (the detection rate) as a function of the number of levels and in terms of the $\rho$ parameter.

## Acknowledgments

Many of the ideas in this paper were obtained by analyzing models developed by L. Zhu and Y. Chen in collaboration with the first author. G. Papandreou and C. Coryell gave very useful feedback on drafts of this work. D. Kersten gave patient feedback on speculations about how these ideas might relate to the brain. The WCU program at Korea University, under the supervision of S-W Lee, gave peace and time to develop these ideas. We gratefully acknowledge support from the ARO 62250-CS and and the Center for Minds, Brains and Machines (CBMM), funded by NSF STC award CCF-1231216.

## References

N. J. Adams and C. K. I. Williams. Dynamic trees for image modelling. *Image and Vision Computing*, 20(10):865–877, 2003.

I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

G. Blanchard and D. Geman. Hierarchical testing designs for pattern recognition. *Annals of Statistics*, 35:11551202, 2005.

E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *European Conference on Computer Vision (ECCV)*, 2002.

M. Chavira, A. Darwiche, and M. Jaeger. Compiling relational bayesian networks for exact inference. In *International Journal of Approximate Reasoning*, pages 49–56, 2004.

J. J. DiCarlo, D. Zoccolan, and N. C. Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.

K. Fukushima. Neocognitron - a hierarchical neural network capable of visual-pattern recognition. *Neural Networks*, 1(2):119–130, 1988.

S. Geman, D. F. Potter, and Z. Chi. Composition systems. *Quarterly of Applied Mathematics*, 60 (4):707–736, 2002.

S.W. Hawking. I think the next century will be the century of complexity. *San Jose Mercury News*, January 23, 2000.

G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–54, 2006.

D. Kersten. Predictability and redundancy of natural images. *JOSA A*, 4(12):2395–2400, 1987.

I. Kokkinos and A. Yuille. Inference and learning with hierarchical shape models. *International Journal of Computer Vision (IJCV)*, 93(2):201–225, 2011.

S. M. Konishi, A. Yuille, J. Coughlan, and S. C. Zhu. Statistical edge detection: Learning and evaluating edge cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25:57–74, 2003.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time-series. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1995.

P. Lennie. Single units and visual cortical organization. *Perception*, 27:889–935, 1998.

J. Mao, J. Zhu, and A. Yuille. An active patch model for real world texture and appearance classification. In *European Conference on Computer Vision (ECCV)*, 2014.

G. Papandreou, L.-C. Chen, and A. Yuille. Modeling image patches with a generic dictionary of mini-epitomes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *Uncertainty in Artificial Intelligence (UAI)*, 2010.

M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025, 1999.

T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29: 411–426, 2007.

S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381 (6582):520–522, 1996.

J. K. Tsotsos. *A Computational Perspective on Visual Attention*. The MIT Press, 1st edition, 2011. ISBN 0262015412, 9780262015417.

L. G. Valiant. *Circuits of the Mind*. Oxford University Press, 2000.

A. Yuille. Towards a theory of compositional learning and encoding of objects. In *ICCV Workshop on Information Theory in Computer Vision and Pattern Recognition*, 2011.

A. L. Yuille, J.M. Coughlan, Y.N. Wu, and S.C. Zhu. Order parameters for minimax entropy distributions: When does high level knowledge help? *International Journal of Computer Vision*, 41 (1/2):9–33, 2001.

M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

L. Zhu, C. Lin, H. Huang, Y. Chen, and A. Yuille. Unsupervised structure learning: Hierarchical recursive composition, suspicious coincidence and competitive exclusion. In *European Conference on Computer Vision (ECCV)*, 2008.

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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.