Computer Vision and Image Understanding 181 (2019) 1–13

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Holistic object detection and image understanding

Gonzalo Vaca-Castano [a,*], Niels DaVitoria Lobo [b], Mubarak Shah [b]

[a] IMEC USA Nanoelectronics Design Center, 190 Neocity Way, Kissimmee, FL 34744, USA
[b] Center for Research in Computer Vision, University of Central Florida, 4000 Central Florida Blvd., Orlando, FL 32816, USA

ARTICLE INFO

Communicated by N. Paragios

MSC:
41A05
41A10
65D05
65D17

Keywords:
Computer vision
Image representation
Object detection

ABSTRACT

This paper proposes a new representation of the visual content of an image that allows learning about what elements are part of an image and the hierarchical structure that they form. Our representation is a Top-Down Visual-Tree, where every node represents a bounding box, label, and visual feature of an object existing in the image. Each image and its object annotations from a training dataset are parsed to obtain the proposed visual representation. These images and their parsed tree representations are trained using a Top-Down Tree LSTM (Long Short Term Memory) network. The encoded information, allows integrate object detection and image understanding in a single process. The presented holistic object detection is not agnostic to the overall content of the image, and it is influenced by the image composition and the parts discovered. During testing time, from an image, we are able to infer the most prominent type of objects and their locations, the parts of these objects, and having a proper understanding of the image content through the obtained Top-Down Visual-Tree representation output. The accuracy of our object detection process increases notably respect to the baseline Fast R-CNN method in the visual genome test dataset.

1. Introduction

Humans derive a great deal of information about the world through their visual sense. Vision accounts for two-thirds of the electrical activity of the brain when the eyes are open (Fixot, 1957). In spite of the recent success in computer vision on individual tasks such as image classification, object detection, image segmentation, and the progress on scene understanding, researchers still lack clarity about computer comprehension of the content of the image as a whole. While most recent efforts have involved the use of language to achieve comprehension (Vinyals et al., 2015; Wu et al., 2015; Fang et al., 2015; Hendricks et al., 2016; Jia et al., 2015; Mao et al., 2016; You et al., 2016; Zhou et al., 2016; Wu et al., 2015, 2016), this paper presents a pure visual representation of the image content that allows an understanding of the image content through a unified framework for object/stuff detection, and representation of the discovered objects, their relations, and their relative importance.

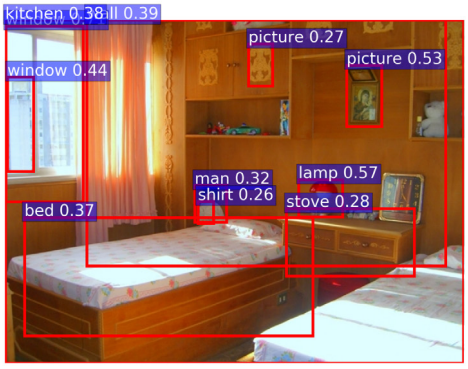
The object detection problem focuses on identifying a particular object from multiple categories. Given a set of window proposals, a multi-label classifier determines scores for the different types of objects and the background in the selected region. Each category competes with others for the highest score that determines the label of the region. In scenarios constrained to a few categories (e.g. less than 200) the described approach works reasonably well; however in more realistic scenarios, a larger number of categories is necessary in order to have a better understanding of the diverse content of images. This increase in the number of categories results in many simultaneous possible valid detections, which makes scene interpretation more difficult. In addition, many categories with weak detectors do not generate high scores, even in scenes where the global and local context strongly suggests the opposite. An example is shown in Fig. 1b where object detections for most prominent classes are displayed. Neither the "drape" nor the "pillow" are detected directly, however these objects could be detected using the context information from the detection scores of the nearby objects "window", and "bed", which are generally highly correlated with the un-detected objects. Hence, the object detection problem with multiple categories will benefit from the incorporation of contextual cues such as object co-occurrence, and overall image consistency.

Images contains more information than just objects. A typical image can be interpreted as a hierarchical structure that relates objects to each other. Starting from the higher level that represents the full image, there are elements that are more prominent or meaningful from a visual and/or semantic point of view. Similarly, each one of these elements may also be related with other elements that have lower relative importance in the image. In the case of Fig. 1a, the elements in the scene in order of their importance are "bed", "window", "desk", and "wall". Each one of these objects has associated other less relevant elements. The "pillow" and the "doll" are associated with "bed". The "drape" is associated with the "window". Similarly, there are some

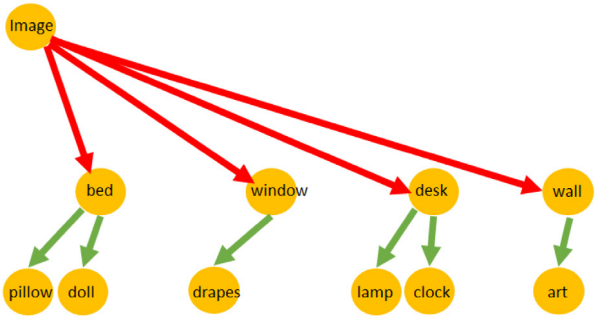
* Corresponding author.
E-mail address: gonzalo@knights.ucf.edu (G. Vaca-Castano).

https://doi.org/10.1016/j.cviu.2019.02.006
Received 30 July 2018; Received in revised form 3 February 2019; Accepted 18 February 2019
Available online 26 February 2019

(a) Image



(b) Most confident detector outputs



(c) Top-Down Visual-Tree (TDVT) image representation. Part dependencies are depicted in green color, and Weak dependencies are depicted in red color. Subsection 3.1 gives details about the representation and explains the types of dependencies .

Fig. 1. An image, the most confident detector outputs, and the TDVT image representation of its content.

"pictures" associated with the "wall". Fig. 1c depicts a proposed tree that captures the inherent hierarchical structure of Fig. 1a. In this paper, we propose a Top-Down Visual-Tree (TDVT) image representation that allows to encode the location of the elements in the image and their hierarchical structure. The proposed representation also encodes the order of importance of the elements and defines their type of relations.

As a first step, we show how to automatically obtain a Top-Down Visual-Tree (TDVT) image representation from object annotations through an image parsing process that operates on available image datasets (with annotations). Then, in the training process, we learn models for object detection and use a novel Top-Down Tree LSTM network to learn the hierarchical structures associated with images from their TDVT image representation. During testing, given an input image, our method generates a rich set of outputs, beyond the object locations. In particular, we are also able to predict the structure of the image, and identify its main salient elements; recognize which objects are parts of others, and identify relationships between objects.

The inference process finds un-detected elements that traditional object detection is unable to capture, since our approach incorporates the image context during object detection.

The contributions of this paper are summarized as follows: (1) a new TDVT image representation that captures hierarchical structure, importance of objects and type of dependencies, (2) a parsing method to obtain the image representation from an annotated dataset, (3) an algorithm to train and learn the visual structure of the images in the dataset, (4) an algorithm to perform inference from an image to detect the objects, degrees of importance, and its hierarchical structure.

2. Related work

The object detection problem has been one of the core problems in computer vision since objects are a key building block of any image. A decade ago, object detection technology started to solve real problems with the finding of visual features like the Histogram of Gradient (HOG) that in conjunction with Support Vector Machines (SVM) provided the reference method (Dalal and Triggs, 2005) for the period from 2005 to 2008. The Pascal VOC challenge (Everingham et al., 2010) was the competition that fueled the object detection research from 2005 to 2012. With a dataset of twenty different object classes, and up to approximately five thousand images, the competition was dominated by methods based on Deformable Part Model (DPM) (Felzenszwalb et al., 2010). The availability of massive image datasets (Deng et al., 2009; Lin et al., 2014a) resulted in deep learning based methods as the current predominant technique for object detection (Girshick, 2015; Ren et al., 2015). The number of available training images in the new datasets increased to hundreds of thousands, and also the number of categories to detect. In the Large Scale Visual Recognition Challenge (ILSVRC) 2016[1] the classes went up to 200 categories, while the recent COCO dataset[2] contains 80.

The core idea of the current dominant algorithms for object detection is the use of Regions of Interest (ROI) that could contain an object, and evaluation of each region to determine if it belongs to a class or to the background. Hence, a detection network typically contains a stack of Convolutional Neural layers (computed once per image), a ROI pooling method that extracts features for each ROI, and a classification network that determines the label of the region. Region of Interests (ROI) can be generated externally by a generic object proposal method (Uijlings et al., 2013; Zitnick and Dollàr, 2014). Most recently, the ROI generation process was integrated in the network (Ren et al., 2015; Redmon et al., 2016; Liu et al., 2016), saving extra computations by re-utilizing the computed CNN features. Further performance improvements has been achieved mainly by changes in the network structure toward the use of deeper networks (Simonyan and Zisserman, 2015; He et al., 2015). The progress in object detection technology have shifted the challenge from tens of possible object categories to thousands of them.

Previous to the emergence of the neural networks as the dominant computer vision tool, many efforts focused on the analysis of context and its relation with object detection (Galleguillos and Belongie, 2008; Divvala et al., 2009; Choi et al., 2010; Song et al., 2011). Galleguillos and Belongie (2008) presented a survey that identifies three types of context, semantic, spatial, and size context; acting at two levels, global and local; and showing two mechanism of integration of the context information. They demonstrated that contextual information can help to successfully disambiguate appearance inputs in recognition tasks. In Divvala et al. (2009), the authors presented an evaluation of the

[1] http://image-net.org/challenges/LSVRC/2016/.
[2] http://mscoco.org/.

role context plays in the object detection task using the Pascal VOC 2008 dataset. Understanding the context as any information source that may influence the way a scene and the objects within it are perceived, ten different sources of context were identified, and experiments were performed with a subset of context sources: local pixel context, 2D scene gist, 3D geometric, semantic, geographic, photogrammetric, and cultural cues. A Bayesian formulation for the object location, size, and presence, and their combinations, was used to improve the object detection based on these cues. The work of Choi et al. (2010) introduced the SUN dataset, and presented a Tree-structured contextual model. Their model is a single graph that represents positive and negative co-occurrences per dataset. A major weakness of their model is that it only captures generalities about the specific dataset. Song et al. (2011) proposed an iterative contextualization system, with the objective of boosting object detection and image categorization by taking the outputs from one task as the context of the other one using a SVM formulation and dynamically adjusting the classification hyperplane. The key of the hyperplane formulation is that context is activated to be supportive mostly for ambiguous samples where context is helpful.

The role of local context in object detection using deep neural networks seems less important than in traditional object detection, presumably due to longer field of view covered by the convolutions. Nevertheless, the context was shown to have a positive impact for object detection using deep neural networks (Bell et al., 2015; Gidaris and Komodakis, 2015; Li et al., 2016; Vaca-Castano et al., 2016). Gidaris and Komodakis (2015) presented a multi-region deep CNN that selects different regions around the ROI to obtain a more robust feature vector. They improved object detection results on the Pascal VOC dataset. Similarly, Bell et al. (2015) form a descriptor that includes vectors from other multiple layers. They used spatial Recurrent Neural Networks (RNNs) to gather local contextual information from above, below, left, and right of the object. They used the descriptors to obtain improved scores and adjusted ROIs. The work of Li et al. (2016) proposed a network that exploits contextual information for object detection using two sub-networks. An attention-based sub-network allows to use a global context view, while a multi-scale sub-network captures local context by pooling descriptors from three bounding boxes scaled around the ROI. Their results were reported in PASCAL dataset.

Latterly, most of the efforts in regards to image understanding have proceeded to the use of Natural Language Processing (NLP) in conjunction with the extra help from visual information. Related problems like image captioning (Vinyals et al., 2015; Wu et al., 2015; Fang et al., 2015; Hendricks et al., 2016; Jia et al., 2015; Mao et al., 2016; You et al., 2016) and visual questioning answering (Zhou et al., 2016; Wu et al., 2015, 2016) have emerged producing impressive results.

The common trend in these methods is that they are based on directly using language to fill the visual semantic gap. Studies with people born completely deaf (Sacks, 1989) related to the way they think in terms of their "inner voice", has shown that, persons born completely deaf, that only learned sign language, will think in sign language. There is no doubt that language is needed in order to have an abstract thinking and self-awareness. We believe, however that a pure visual level representation of the image content is still missing in the current trending abstraction, since it jumps directly from objects and global descriptors to the language, without a notion about the overall content of the image. Hence, in this paper, we propose a novel purely visual hierarchical representation that allows modeling the image content, and learning about the image structure from datasets with massive amount of images. The proposed representation and learning strategy is utilized to perform object detection using thousands of categories while simultaneously understanding the overall content of the image, determining the relative importance of their objects, and identifying their hierarchies.

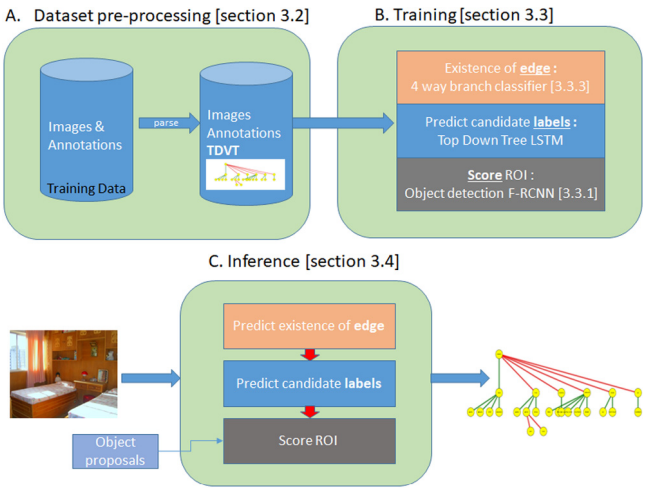


Fig. 2. Parts of the proposed framework for object detection and image understanding.

3. Framework

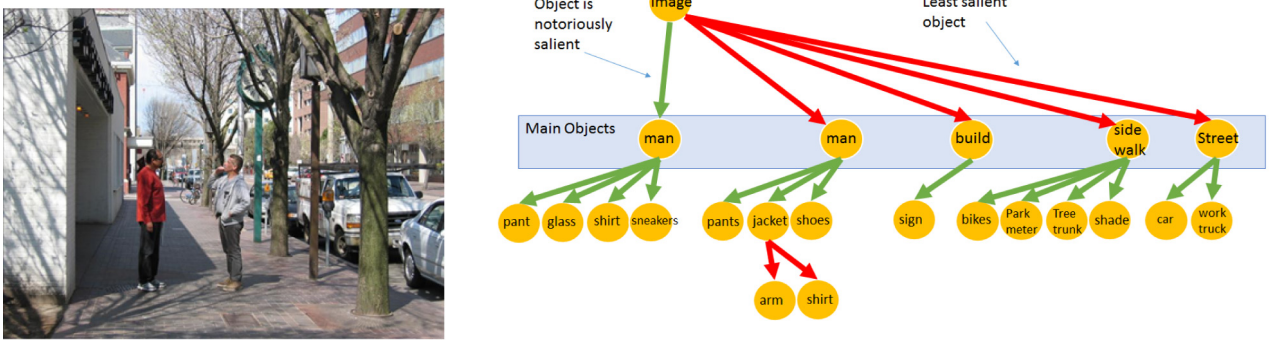
In this section, we describe in detail the proposed framework. Our goal is to provide a holistic description of the image, that accounts for the presence of elements (object detection), identification of the image's main salient elements, and finding the relationships between objects that form a hierarchical structure (see Fig. 2).

Our TDVT image representation describes the content of the image as a hierarchy of objects encoding their importance and type of relation. Section 3.1 provides details about this representation. Section 3.2 is dedicated to the pre-processing stage that generates TDVT training data from the images and annotations of existing datasets. Then, in Section 3.3 we describe our training process, that learns a knowledge base about the objects and the image composition from the training images and the TDVT representations previously generated. The training phase includes the learning of three different parts from a tree. They are: (a) the existence or absence of an edge in a tree, (b) the node label (object category) associated to an existing edge, and (c) the score associated to the visual information of a node the tree. Finally, in Section 3.4 we provide the details about the inference process where the hierarchical structure and the objects are discovered. In the inference, we repeat a sequential process that includes determining the existence of an edge, predicting the possible labels for the object linked to the edge, and scoring the candidate regions of interest (ROI).

3.1. Image representation

Representations are important in order to model information about the world in a form that a computer system can utilize to solve complex tasks. Consider an image in Fig. 3a. If any person is asked to describe the content of the image, a very fair approximate description would be something like this: "there are two men on a sidewalk next to a building and the street ". Just as in this particular case, We will describe images based on the entities that are part of the image. Beyond this, each one of the described entities can also be described by their constituent parts and relations with other entities. In the same image, for example, the man on the left side wears pant, shirt, glasses and sneakers. Hence, we observe that an image can be described properly as a hierarchical structure of related objects.

A natural data structure to represent hierarchical constructions is the tree. A tree is a collection of nodes starting at a root node, where the nodes are connected to other nodes through edges, without having any cycle (sub-tree children have only one parent node). Every node is a data structure that contains associated information. In the case of an image, we represent every node as a possible object/stuff/attribute



(a) Image

(b) Top-Down Visual-Tree (TDVT) image representation. Part dependencies are depicted in green color, and Weak dependencies are depicted in red color.

Fig. 3. Image and its proposed TDVT image representation obtained from its annotations.

and its associated information. The information associated to each node includes the bounding box location within the image, identity (class) of the object, and feature vector. The root node corresponds to the full image.

In order to describe the image visually, it is important to encode the relative importance of the objects. Not every object in an image is equally important. In case of Fig. 3a, the most prominent object in the image is the man in red shirt. After that the man wearing a jacket captures our attention next, followed by the building, sidewalk, and street. Visual information has an inherent sequential ordering in the sense that some objects capture our attention more quickly than others. Our image representation captures this property through the ordering of the siblings of any sub-tree. As can be appreciated in Fig. 3b, the child branches that are more important are listed first (closest to the left), while less salient children are listed at the end. As we will show further, the ordering is important since most prominent objects are discovered first in the inference process, resembling attention models.

Finally, in our TDVT representation, we distinguish between two types of child edge relations. The first one, depicted as green edges in Fig. 3b corresponds to relations that imply clear ownership dependencies. This applies to the case of object and parts, or parts and some inherent properties such as color or shape. The second type of childhood relation, depicted as red edges, model weaker relations between the nodes where some dependence exists, but there is no clear ownership implied. For the rest of this paper, we will denote the first type of relations as Part dependencies, and the second type of relations as Weak dependencies. Examples of Part dependencies in the figure are man–jacket, man–pants, man–shoes where all the siblings correspond to parts associated with the man. An example of Weak dependence is jacket–arm, where ownership association between jacket and arm is not so evident. A special case in the definition of Part and Weak branches is at root level. In this case, a Part dependence represents an object that is notoriously salient with respect to other objects in the image. In the example, the man in red is considered a Part dependence, and all the remaining main objects are part of the Weak dependence.

3.2. Parsing images from a dataset

The proposed TDVT image representation allows to describe the content of the image in a hierarchical manner, encoding the importance of objects, and relations between the objects. The TDVT representation becomes powerfully useful when we are able to represent a large number of images of which we can learn generalities about the image composition. Annotation of a complete dataset is a titanic task due the size of the current datasets (more than 100k images). We are interested in avoiding a special annotation process of the training image in the dataset, since the associated high costs, and in order to keep our approach as independent as possible from the annotation process. Additionally, obtaining consensus from different annotators could be cumbersome, since there is not a unique way to build the proposed tree representation. Therefore, it makes more sense to create the tree image representation from existing annotations and set up a set of parsing rules to build tree representations in an unified way.

Assuming that the training dataset contains bounding box annotations of the possible objects/stuff present in the image, we describe below the five steps involved in parsing a large image dataset to obtain a TDVT representation. Fig. 4 shows the parsing obtained for two images randomly selected from the training dataset.

3.2.1. Merging duplicate instances from annotations

Our assumption is that we are dealing with a large number of classes, annotated by humans (typically turkers). Annotation corresponds to a bounding box b_i that enclose an object annotation i, and has a label l_i. Typically, multiple individuals work on the same image, and it is quite common to find bounding boxes approximately enclosing the same object with different labels. In this step, we eliminate the duplicated annotations by considering the overlap between all the annotated bounding boxes for the image. We use the Jaccard index as a measure of the similarity between two bounding boxes b_i and b_j. Jaccard index between bounding box b_i and b_j is defined as:

$$J(b_i, b_j) = \frac{\left| b_i \cap b_j \right|}{\left| b_i \cup b_j \right|}, \tag{1}$$

where the numerator is the area of the common region enclosed by the bounding boxes b_i and b_j, and the denominator is the total area of the union of the regions enclosed by bounding boxes b_i and b_j. The bounding boxes with Jaccard index close to one,

$$J(b_i, b_j) > 1 - \gamma, \tag{2}$$

encloses approximately the same region of the image.

Annotations that satisfy the last condition are then examined semantically to determine if the annotations are variants of the same object. In order to examine the semantic similarity, we use the word distance between the labels of the annotated objects. The distance is computed using a path-based distance measure for words from the wordnet (Miller, 1995) hierarchy. Two labels might describe the same object if they have high semantic similarity,

$$\left| l_i, l_j \right|_s > 1 - \epsilon, \tag{3}$$

where the score for a perfect match $l_i = l_j$ has a score equals to 1. Instances with high semantic similarity (Eq. (3)) and similar bounding boxes (Eq. (2)) are merged in a single node. They are considered noisy annotations of the same object.

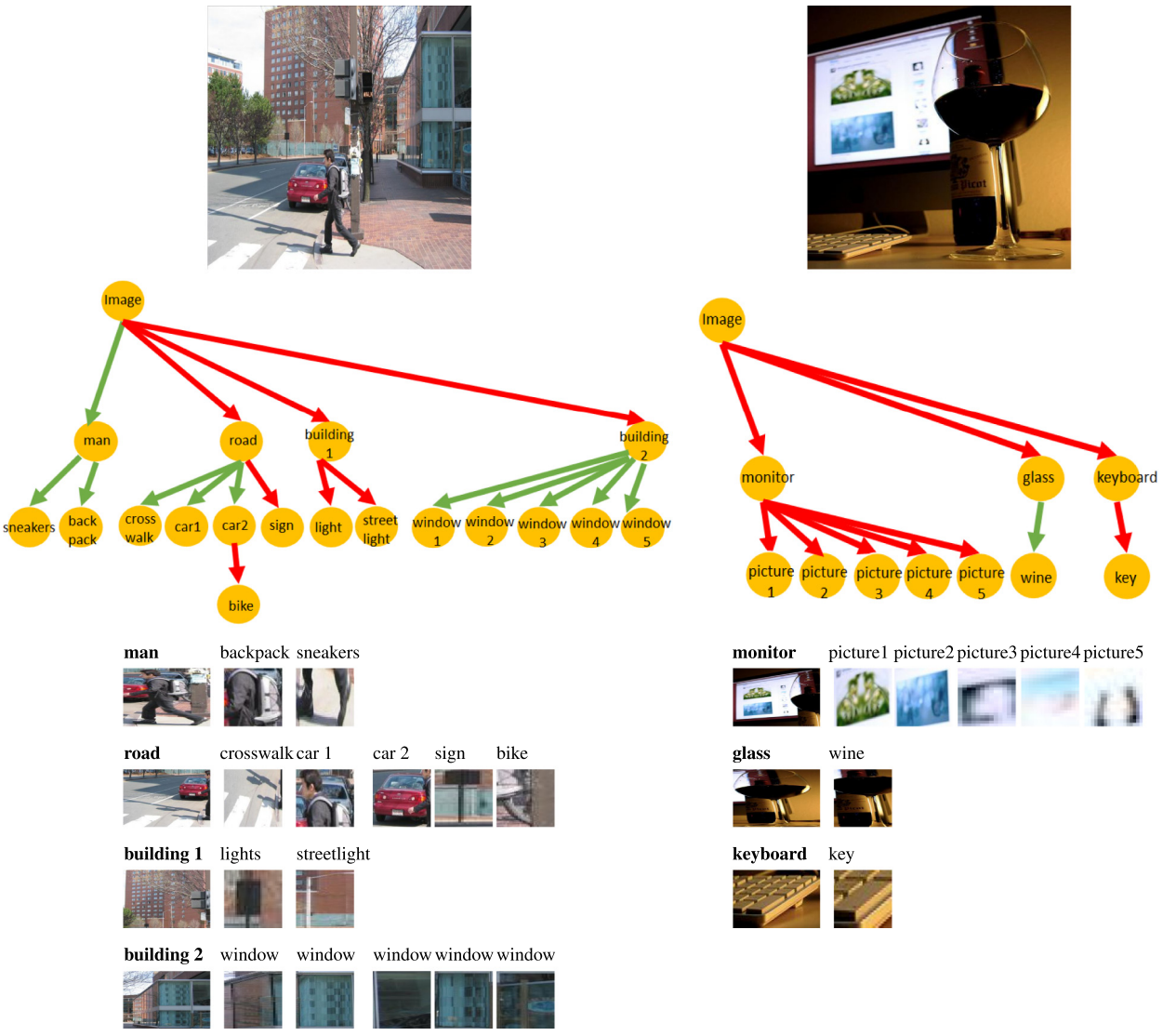


Fig. 4. Examples of two images (at top) that were parsed from the dataset. Their TDVT image representations are depicted in the middle, while the images on the bottom are the different objects annotated that are part of the final Tree representation. Part dependencies are depicted in green color, and Weak dependencies are depicted in red color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.2.2. Connecting related nodes

Our objective is to build a tree from the existing bounding boxes and labels. Every annotation available after merging duplicates (previous step) is a possible node of the tree $\mathbf{V} = \{V_1, V_2, \ldots, V_j, \ldots, V_N\}$. In this step, we define the mechanism to create edge connections, between the available nodes, $E = \{\{V_j, V_k\}, \ldots, \{V_l, V_m\}\}$. The main assumption here, is that nodes that are related must have at least some degree of visual overlapping. Hence, we only consider connections between nodes with Jaccard index greater than zero.

$$\{V_j, V_k\} \in E, \quad \text{if } J(b_{V_j}, b_{V_k}) > 0. \tag{4}$$

Each one of these edges are analyzed individually to determine the type of dependence (Part or Weak) between the related nodes.

An obvious Part dependence exists when one of the nodes (child) is inside other node (parent), and there is a close semantic similarity between the nodes.

$$\{V_j, V_k\} \in E^P,$$
$$\text{if} \quad b_{V_j} \cap b_{V_k} > \zeta \cdot \left|b_{V_k}\right| \quad \wedge \quad \left|l_{V_j}, l_{V_k}\right|_s < 1 - \epsilon_2, \tag{5}$$

where E^P is the set of Part dependencies, $\left|b_{V_k}\right|$ is the area of the bounding box b_{V_k}, ζ is a tolerance value for the area intersection, $\left|l_{V_j}, l_{V_k}\right|_s$ is the semantic similarity between the nodes V_j and V_k, ϵ_2 is a tolerance value for the word similarity, V_j is the parent node, and V_k is the child node.

There are datasets that have extra annotations that can be utilized to define dependencies. For example, the visual genome dataset (Krishna et al., 2016) provides relations between nodes through word connections that can be used to define edges between nodes. The existence of Part dependencies can be established by matching word connectors against a list of key words that denote ownership relation such as "in", "wears", and "have".

Any edge that belongs to E, which is not a Part dependence E^P is considered to be part of the set of Weak dependencies E^W,

$$E^W = E \setminus E^P. \tag{6}$$

3.2.3. Generating sub-trees

The obtained list of edges E is used to identify formed disjoint sub-graphs. In this step, each one of these sub-graphs are converted into a tree structure.

The first characteristic of a tree is that the edges do not form any cycle. However, the disjoint graphs obtained so far, do not have this property. A spanning tree of a graph is just a subgraph that contains

all the nodes without cycles. A graph may have many spanning trees, but the Minimum Spanning Tree (MST) is an algorithm that connects all the nodes together with the minimal total weighting for its edges. Hence, any existing loop in each graph is removed through a Minimum Spanning Tree (MST) algorithm, where the weights of the edges in the graph are defined using the path-based distance measure for word similarity from wordnet hierarchy,

$$w(\{V_j, V_k\}) = \left| l_{V_j}, l_{V_k} \right|_s. \tag{7}$$

Each one of the obtained group of nodes that do not form loops are analyzed to identity their root nodes, and edge directions are defined using parenthood hierarchical relations. The parenthood relations from Part dependencies $E^P = \left\{ \left\{ V_{\text{parent}_i}, V_{\text{child}_j} \right\}, \ldots, \left\{ V_{\text{parent}_n}, V_{\text{child}_o} \right\} \right\}$ automatically determines their edge directions.

Any node in a tree must have at most one parent. In cases where the existing Part dependencies indicates that a node has multiple parents, only one parent edge is selected,

$$E^P = \{\{V_i, V_j\}, \{V_i, V_k\}, \{V_i, V_z\}, \ldots, \{V_x, V_y\}\} \Rightarrow$$
$$E^P = \{\{V_i, V_j\}, \ldots, \{V_x, V_y\}\}, \tag{8}$$

using the largest area overlap between the node and their possible parents is utilized as criteria to determine parenthood relation,

$$V_i \cap V_j > V_i \cap V_k, \ldots, V_i \cap V_z. \tag{9}$$

A list of root node candidates is created from the existing Part and Weak dependencies. A node with only one connection and relative small bounding box size is a leaf, and then removed from the list of root node candidates. Nodes are examined one by one starting from known leaves until only one root node remains in the list. The criteria used for discarding nodes are (a) consistency in the sizes of the bounding boxes which allows to identify directions in Part Dependencies, and (b) all the nodes can have at most one parent. The last condition implies that if a parent exists for a node, any other edges connecting the same node are children connections and are discarded from the root node candidate list.

3.2.4. Building a preliminary tree

The set of resulting sub-trees are gathered to form a preliminary tree. The root node is defined as the node covering all the image. Each one of the root nodes from the generated sub-trees becomes the main objects of the image representation (See Fig. 3b). Nodes that are isolated are considered irrelevant and not included in this preliminary visual tree representation.

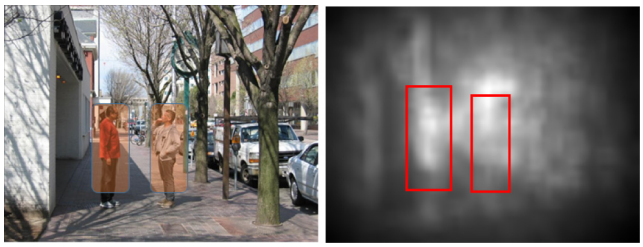
3.2.5. Re-ordering branches

The preliminary visual tree was built without any consideration about the importance of the nodes of the tree. In this step, each branch of the tree is analyzed independently. The nodes with a common parent are sorted according to their visual importance (see Fig. 5).

A saliency map is a function that gives a subjective perceptual value about the property that "makes some items in the world stand out from their neighbors and immediately grab our attention".[3] Given a saliency map, we define the saliency density for a region of interest as the sum of the saliencies values enclosed by the associated object bounding box divided by its area. Saliency density is computed for every node of the tree.

Every branch is then independently processed. The nodes with a common parent are sorted in a descending order according to its saliency density value. Finally, a special treatment is operated over the children of the root node to identify if the nodes qualify as Part or Weak dependencies. Nodes with a very high saliency density values are interpreted as Part dependencies and the rest of the nodes becomes Weak dependencies.

[3] Laurent Itti (2007), Scholarpedia, 2(9):3327. http://www.scholarpedia.org/article/Visual_salience.



(a) Image and two of its regions of interest

(b) Saliency Map and the regions of Interest used to compute saliency density

Fig. 5. Image and saliency map used to compute the visual importance.

3.3. Training

Each node of the TDVT representation is formed by a label, a bounding box and the possible connections to other nodes. These three aspects of our representation are learned separately, but assembled together on inference time. Now, we provide details about these three training stages.

3.3.1. Learning to score ROI

The extraction of the visual information and scoring of the bounding boxes for the different labels is performed using Fast R-CNN architecture (Girshick, 2015). Fast R-CNN architecture takes an image as input into a Convolutional Neural Network(CNN), where Multiple Regions of Interest (ROI) are evaluated one by one using a ROI pooling layer that maps the outcome of the CNN spanning the ROI into a fixed-size vector. Each one of these vectors pass through a multi-layer Perceptron (MLP) that outputs the scores for any possible object label. We trained a fast-RCNN network, and use the layers and weights learned to process visual inputs of our system.

3.3.2. Learning to predict objects

The core part of our system, is a network that combines the visual layers trained on Fast R-CNN and a neural network architecture named Top-Down Tree LSTM (Long Short-Term Memory) (Zhang et al., 2016), to predict the most probable type of object (label) that could be associated to a node, given the already seen tree structure of the image and its visual information. The composition of this network is presented in the Fig. 6. The inputs for the training process are an image and its TDVT image representation. Visual features are extracted for the image using the CNN layers and the ROI layer learned from the trained Fast-RCNN model. Each node of the TDVT representation has associated a bounding box that is used as Region of Interest to extract visual features.

Starting from the root node of the TDVT representation, the Top-Down Tree LSTM building block learns, edge by edge, to predict the category (label) of the node linked by the edge currently trained. The nodes and their edges are processed sequentially in a Breadth-first order. The predicted output w is encoded as a one-hot vector. The vector consists of 0s in all components with the exception of a single '1' , in a component used uniquely to identify the predicted label. The fully connected (FC) layer at the end of the network uses as input the hidden state from the recurrent neural network (RNN) unit, and outputs the vector w after passing for a softmax function.

The network is trained with Noise Contrastive Estimation (NCE) loss (Amnih and Teh, 2012). As in any other recurrent neural network (RNN), the sequential signal is fed one by one to the network. Training is done in mini-batches of size b. In step 1 of the training, the first edge associated to the root of the b images is input in parallel. In step 2, the second edge associated to each one of the b images is input in parallel.

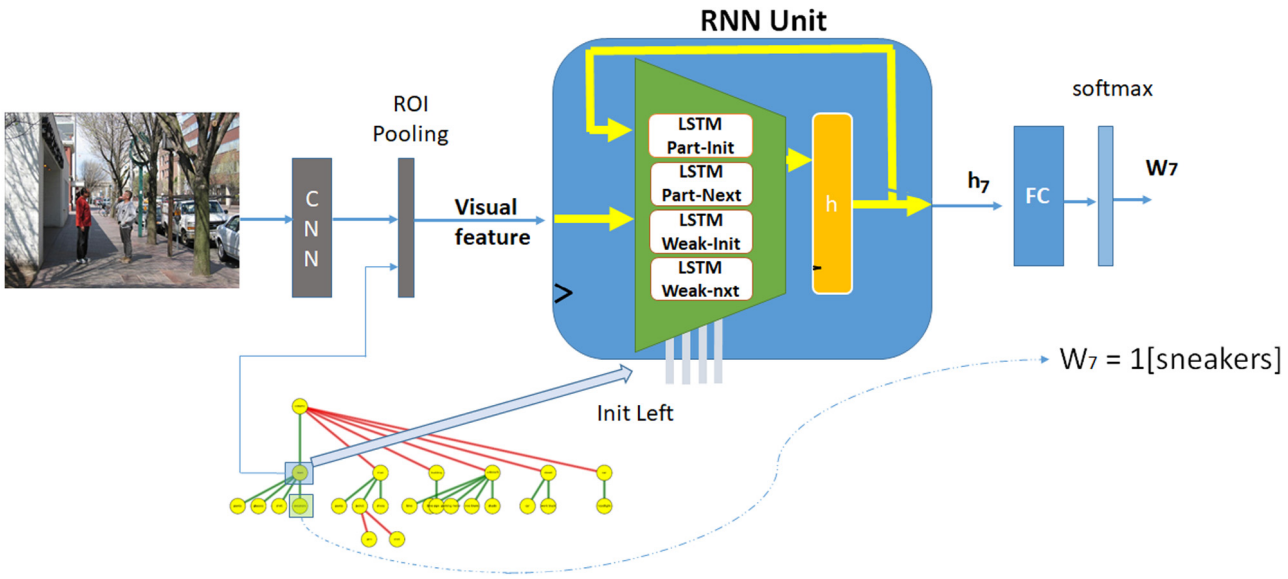


Fig. 6. Proposed training model utilized for the prediction of the associated object according to a hierarchical structure. The input image passes once through a Convolutional Neural Network, and visual features are extracted for each node of the tree structure representation. The network is trained with the goal of predicting the label of the node that the examined edge is connecting to. In the figure, the edge that contains as a starting node the man in red shirt is being examined, and the edge must predict the correct label 'sneakers'. The predicted label, W_7, is encoded as a one-hot vector. Every new processed edge updates the value of the hidden state h_7 (the '7' makes reference to the sequence ordering). The hidden state keep tracks of the tree structure previously processed.

The mini-batch is finalized when all the edges of the b images are fed. Each TDVT tree in a batch is handled in parallel, but the network only sees one predicted output word from an edge at a time and does the computations accordingly.

3.3.3. The recurrent neural network (RNN) unit

The RNN unit contains four LSTM units, a shared hidden state vector h, and a tree dependency selector that controls which LSTM unit is activated. From the four LSTM units, two of them are only used when there are edges with dependencies of type Part, and the other two only take care of the edges with dependencies type Weak.

Part dependencies are modeled by a first LSTM that control initial Part dependence (Part-Init), and a second LSTM unit that controls the connections between the remaining nodes of the same parent (Part-Next). Similarly, Weak dependencies are controlled by two LSTM. One unit controls the initial Weak connections (Weak-Init), and the other unit controls consecutive Weak dependencies of the same parent node (Weak-Next).

The input to the unit are $\boldsymbol{x_t}$ and z_t. $\boldsymbol{x_t}$ denotes the visual feature pooled from the node, and $z_t \in \{Part-Init, Part-Next, Weak-Init, Weak-Next\}$ denotes the type of edge dependence of the currently analyzed edge. Let $\boldsymbol{H} \in \Re^{d\times(n+1)}$ represent the shared hidden states of all the existing nodes, where d is the hidden unit dimension size, and n the number of nodes in the image representation. Every time an edge is processed, only one of the four LSTMs is activated based on edge type z_t. The hidden state is updated as:

$$\boldsymbol{h_t} = LSTM^{z_t}(\boldsymbol{x_t}, \boldsymbol{H}[:, t']), \tag{10}$$

$$\boldsymbol{H}[:, t] = \boldsymbol{h_t}. \tag{11}$$

3.3.4. Learning to predict the existence of edges

The network trained in the previous subsection learns to predict what object could attached to an edge that is currently examined. However, during testing, the input is just an image, we do not know if the edge for a particular exist or not. Therefore, every time a node is found, it is necessary to determine if the node has an edge, and what type of edge the node has.

To solve this inconvenient a set of four different classifiers are trained, given that there are four possible type of dependence (Part-Init, Part-Next, Weak-Init, Weak-Next).

The inputs for these classifiers are the visual features of the analyzed node, and the current hidden vector h_t state. The output of these classifiers are 1 or 0, if the particular type of edge exist or not.

3.4. Inference

This subsection describes the procedure to obtain the detected objects and the associated TDVT image representation from a test image. A first assumption, is the availability of object proposals for the image, which are obtained using some of the existing object proposal methods. The TDVT image representation is built gradually edge by edge. Then the inference process starts from the root node which represents the full image. The convolutional layers from the model trained for object detection are used to extract visual features. Using the visual feature for the full image and the hidden vector h at its initialization value, the classifier for Part-Init dependence is evaluated to determine the existence of a Part dependence. In the case that the output is positive, the Top-Down Tree LSTM network for the Part-Init dependence is utilized to predict a label vector. This vector indicates how likely is to find each of the object categories in the image. A object detection search is performed in the image for a limited set of object categories determined by the label vector. The search is performed by computing scores using the classification layers of the trained object detector, having as input the visual features pooled from the bounding boxes provided by the object proposal method. The detection with the highest score among most likely objects, is declared as the node connected by the Part edge. Subsequently, the existence of other Part dependencies for the examined node are determined by the classifier for Part-Next dependence. Visual features extracted from the latest found node and the corresponding updated hidden vector h_t are used as inputs to the classifier. Every time a Part-Next dependence is found to exist, the Top-Down Tree LSTM determines the most likely labels for the corresponding edge. The object search is performed in a reduced set of categories, boosting the possibility to find visually low-scoring objects of categories related to the examined visual content; in similar fashion than on Part-Init dependence. After the Part branch is completely processed (or the root node does not have a Part dependence), identical course is enforced for the Weak branch.

A classifier for the Weak-Init dependence determines the existence of a Weak branch. If a Weak dependence exists, then the Top-Down

Tree LSTM determines the possible categories of objects related to the edge. The object search procedure is performed only among the found possible categories. When the classifier for Weak-Next dependencies indicates the existence of an additional Weak dependency, the same process of category prediction and object search in a reduced set of categories, is repeated until there are no more positive Weak-Next dependencies.

input :
- Image to be processed
- Object proposals (from a generic object proposal method)

output: TDVT representation of the image

Initialization;
Initialize NextList to empty;
Initialize h_0;
Get visual feature (full image);
Compute 4-way branches classifier (Section 3.3.4);
If *Part-init* **then**
| Fill up InitList indicating the node and their type of edge is Part-Init;
If *Weak-init* **then**
| Fill up InitList indicating the node and their type of edge is Weak-Init;
while *NextList is not empty OR InitList is not empty* **do**
  while *NextList is not empty* **do**
    Get visual feature (ROI of analyzed node), Get h_i;
    Predict edge existence using the trained classifier (Section 3.3.4);
      Update NextList if there are Part-Next or Weak-Next edges;
      Update InitList if there are Part-Init or Weak-Init edges;
    Predict labels for the edge → set {label name, Label score};
    Pick the top N labels with highest label score;
    From the N selected labels:
      Pick object (proposal, label) with highest det. score (Fast-RCNN) ;
    Add the object as a new node of the tree;
  end
  if *InitList is not empty* **then**
    Get visual feature (ROI of analyzed node), Get h_i;
    Predict edge existence using the trained classifier (Section 3.3.4);
      Update NextList if there is a Part-Next or Weak-Next;
      Update InitList if there is a Part-Init or Weak-Init;
    Predict labels for the edge → set {label name, Label score};
    Pick the top N labels with highest label score;
    From the N selected labels:
      Pick object (proposal, label) with highest det. score (Fast-RCNN) ;
    Add the object as a new node of the tree;
  end
end

Algorithm 1: Algorithm to infer a TDVT representation from a test image.

Once the children for the root node are computed, we proceed to infer edges and nodes of lower levels of the tree using the aforementioned object search process used for Part and Weak dependencies in the root node. Part dependencies deal with parts, therefore it is reasonable to process only the bounding boxes with some level of overlap respect to the parent node. The search process in the Weak dependence is more flexible and only requires that the parent node has at least one pixel of overlap with respect to the parent node. The result of this operation is a tree that represents the image, with nodes that contain object detections results (bounding box, label, and score).

The inference process is summarized in the Algorithm 1. Two lists are used to control which node will be processed next. A list named NextList contains information about nodes that have edges type Part-Next or Weak-Next, and a list named InitList contains information about nodes with edges type Part-Init or Weak-Init.

4. Experiments

We perform experiments on the visual genome dataset (Krishna et al., 2016). The visual genome dataset has 108,077 images from the intersection of the YFCC100M (Thomee et al., 2016) and MS-COCO (Lin et al., 2014a) datasets. The annotation includes 5.4 million region descriptions and 2.3 million pair-wise relationships, which are used as possible nodes of the TDVT image representation.

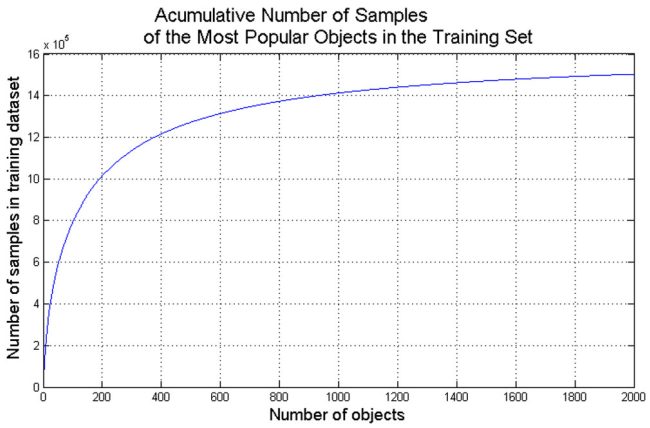


Fig. 7. Accumulative number of samples for the most common objects in the dataset.

Using the training data, the number of instances of each object category after obtaining TDVT representations are sorted in descending order to establish the most common object categories. We selected the first two thousand popular labels as the classes to be used in the experiments. The most common class of the training dataset is 'man' with 37,292 training samples, while the class ranked 2000 is 'desert' and only has 52 samples (see Fig. 7).

Fine-tuning of a Fast R-CNN object detection model is performed using the full annotations of the selected two thousand classes. The network used to train the object is based on the VGG 16 network (Simonyan and Zisserman, 2015) and EdgeBox (Zitnick and Dollàr, 2014) is used as the object proposal method.

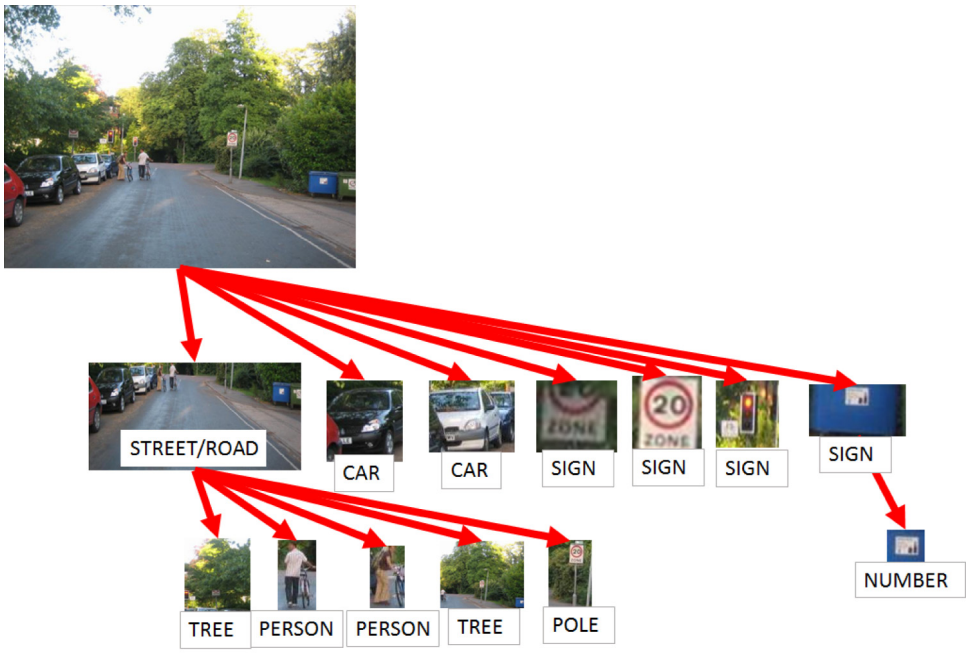
We now describe the choice of parameters and implementation details for the image parsing process of the dataset. We selected the parameters for merging similar objects in an image as $\gamma = 0.2$, and $\epsilon = 0.5$. Edges are considered Part Branches if the intersection of the nodes is more than 80% the area of the smallest bounding box ($\zeta = 0.8$), and path similarity is over 0.3 ($\epsilon_2 = 0.7$). We use Kruskal's Algorithm to solve the Minimum Spanning Tree (MST) and generate trees from graphs. The saliency method by Duan et al. (2011) was used to determine the order within the siblings nodes. A density saliency over 0.8 defines a Part dependence in the root level.

The training details of our network are described next. The Top-Down Tree LSTM of our experiments has a hidden vector dimension of 300, was trained with Stochastic Gradient Descent(SGD), using batches of 64 images. The four classifiers to learn to build the tree was implemented as a Multi Layer Perceptron (MLP) with two hidden units of 300 dimensions.

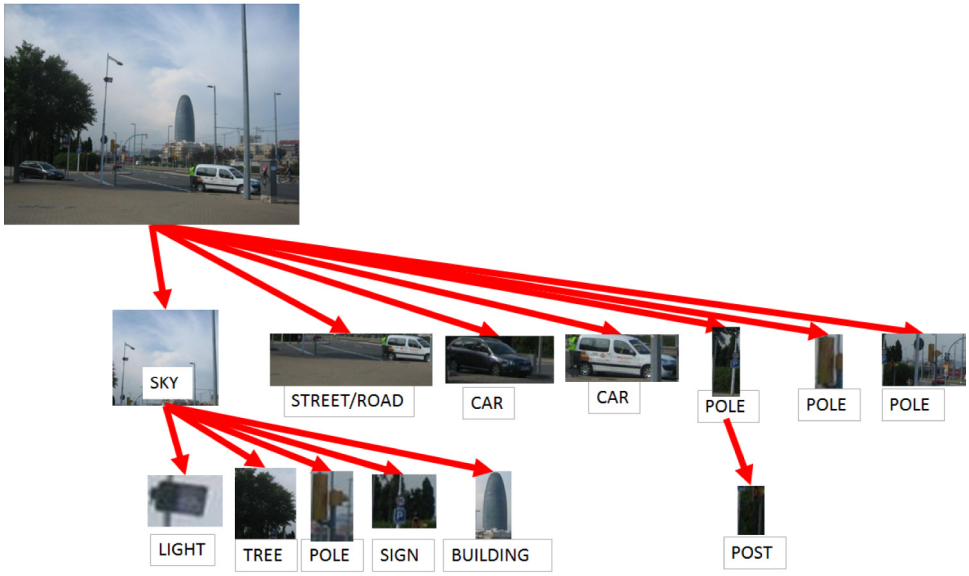
4.1. Qualitative results

In this section we showed some qualitative results of the proposed approach to perform simultaneously object detection and image understanding. Test images are feed to infer a TDVT representation that express the visual content of the image.

Fig. 8 shows the results obtained for two randomly selected test images. While traditional object detection produces a set of bounding boxes and scores with the most likely objects, our method produces a reduced set of objects that are essential to describe the visual content, and the possible relations between these objects. For example, Fig. 8a shows the street/road as the most important objects, followed by two cars, and some signs. Associated to the street/road, in a lower level of importance the image contains two trees, a pole, and two persons.



(a) TDVT image representation for a test image. The main objects in the image in descending order of importance are: road, car 1, car 2, and different signs. The road has associated a pair of trees, a pole, and the two persons at the end of the road.



(b) TDVT image representation for a test image. The main objects in the image in descending order of importance are: sky, street, car 1, car 2, and different poles. The sky has associated objects such as a tree, a pole, a light, a sign, and a building.

Fig. 8. Obtained TDVT image representations for some testing images. Part dependencies are depicted in green color, and Weak dependencies are depicted in red color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 9 shows the object detection results of some testing images. For the proposed method (right side), we show only the detected objects from the children of the root node, which determines the most important objects of the image. For the Fast-RCNN (left side) we show the most confident results according to the object detection score. Our method allows to find un-spotted objects like the "'woman"' in the first set of images, or the "'monitor"' screen in the second set of images. It also, allows to find the most important elements of the images such as the "'street"' and the "'building"' in the third set of images. Finally, our method can get rid of inconsistent categories such as the "'person"' in the last set of images.

4.2. Dataset parsing

We perform a user study to evaluate the quality of the visual tree representations obtained after parsing the images and annotations of the visual genome dataset. Amazon mechanical turkers were

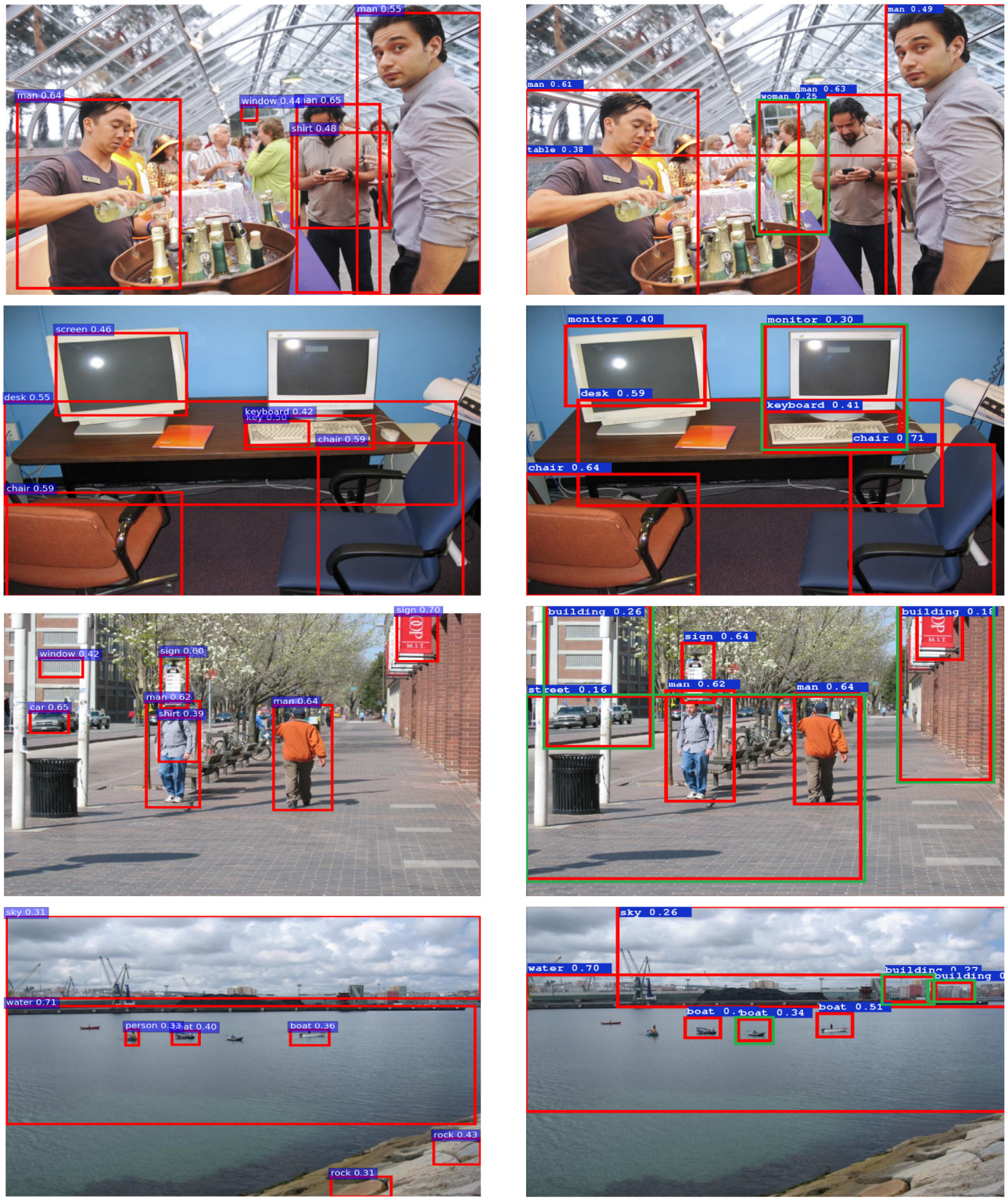


Fig. 9. Examples of images showing the most confident detections using Fast R-CNN object detection (left side) and the proposed method (right side).

used for this evaluation. Every image of the visual genome dataset was presented next to the visual tree representation obtained from the annotations. A total of 156 turkers were asked to tell if the visual tree properly represents the content of the image. Each image was rated by at least seven different turkers. The results are presented in Table 2. Most of turkers considers that the parsed image representations represents properly the content of the images of the dataset.

Table 1
Comparison of the Average Precision–Recall (APR) of Fast R-CNN detector and the proposed approach for the forty four most popular classes of the dataset. The mean Average Precision (mAP) of the two thousand classes is shown in the last column of the table. Our method improves the detection in 38 out of the main 44 classes.

	Man	Window	Person	Shirt	Wall	Ground	Building	Sign	Tree	Woman	Light	Head	Pole	Grass	Hair
FRCNN	42.91	31.21	17.39	42.69	18.99	28.02	35.28	47.43	35.82	**36.02**	17.03	40.75	15.82	21.84	30.56
Ours	**50.38**	**42.75**	**24.23**	**45.47**	**22.93**	**29.03**	**44.04**	**49.93**	**44.13**	31.28	**19.58**	**42.60**	**20.09**	**30.42**	**32.97**
	Hand	Sky	Table	Water	Leg	Car	People	Pants	Clouds	Wheel	Eye	Ear	Door	Hat	Trees
FRCNN	30.26	36.12	**34.82**	42.33	20.91	45.75	24.59	**39.04**	6.77	25.02	19.04	22.17	18.47	**21.22**	16.60
Ours	**33.26**	**36.19**	33.09	**48.18**	**31.18**	**47.77**	**32.92**	38.21	**18.11**	**29.68**	**19.39**	**26.35**	**21.95**	20.89	**21.93**
	Floor	Plate	Line	Shadow	Leaves	Snow	Nose	Shoe	Jacket	Chair	Tail	Fence	Windows	Letter	**Mean**
FRCNN	26.20	**45.42**	11.93	4.36	10.95	24.03	16.45	21.93	**33.43**	41.58	23.51	15.62	3.80	15.23	11.12
Ours	**31.69**	43.27	**17.00**	**6.31**	**20.48**	**27.14**	**20.15**	**26.85**	32.66	**42.20**	**25.20**	**20.19**	**7.27**	**24.26**	**22.53**

Table 2
Evaluation of the parsing of images from annotations. 86.83% of the users considers that the obtained tree represents properly the content of the image.

Yes	No
86.83%	13.17%

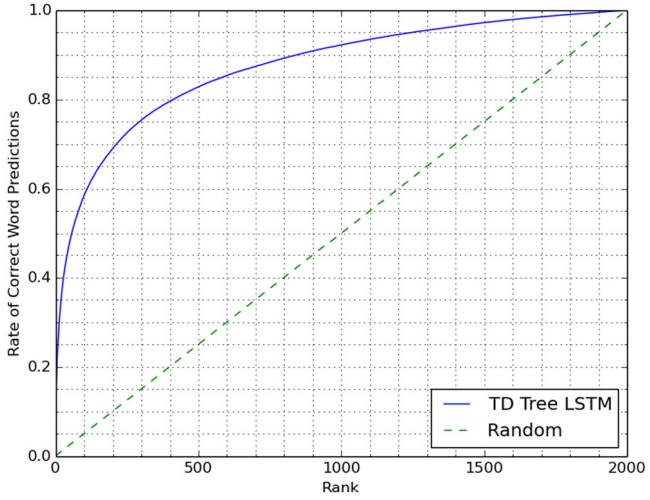


Fig. 10. Rate of correct prediction of categories for different ranks.

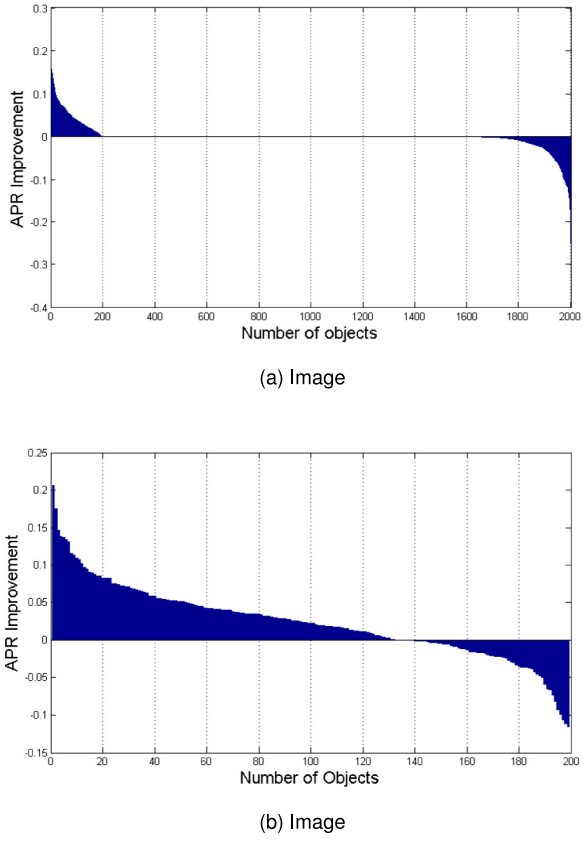


Fig. 11. Improvement of our model over the baseline Fast R-CNN. Object categories are sorted by the improvements. Figure (a) shows the full 2000 categories, Figure (b) shows the 200 most common categories. The overall improvement is achieved by a high improvement in the most popular categories in the training dataset.

4.3. Category prediction

The trained network predicts the category of the object linked by the edge currently examined, given the sub-tree processed until that point. Hence, a prediction is performed for each existing edge during inference time. The test dataset of the visual genome dataset is parsed to obtain TDVT representations of the test images (previous subsection). We use these representations as ground truth and evaluate the performance of the category prediction task for each edge. We evaluate the quality of the predictions as a rank problem using a Cumulative Match Characteristic (CMC) curve. Fig. 10 shows the results.

The prediction process allows to guide the search process. Our framework predicts the right category in 60% of the cases, when the top 100 most likely categories are considered. Clearly, our framework is a lot better than giving to all the categories the same chance as in traditional object detection.

4.4. Improved object detection

We present quantitative results performed over the 5000 images from the original test split of the visual genome dataset.

We compare our method against the object detection results of the Fast-RCNN object detector. The Table 1 shows the Average Precision–Recall (APR) for the forty four most common categories in the training dataset. Our method improves the detection in 38 out of these 44 classes. The average improvement of this large set of categories is +4.89 APR per category, while the loss in performance of the remaining six categories is considerable lower (−1.75 APR per category). The loss in performance in the six classes was small compared to the gain on most of them.

The last column of Table 1 is the mean Average Precision (mAP) computed considering all the objects in two thousand object categories. The mAP for the Fast-RCNN detector with two thousand categories was 11.12, and it increased to 22.53 using the proposed approach.

Fig. 11 shows the relative improvements of the two thousand classes that are detected sorted by the improvement in the APR. Fig. 11a shows that the number of improved categories is lower than the number of categories that decrease their APRs when all the two thousand categories are considered. If we only consider the two hundreds most popular categories from the training dataset (Fig. 11b), we notice that

Table 3
Evaluation of the object detection results in the test set of SUN2012 dataset trained with the data of the visual genome dataset. Metric used is mAP.

Fast R-CNN	Ours
3.40	**4.12**

Table 4
Evaluation of the obtained TDVT image representation. 85.48% of the users considers that the obtained tree represents properly the content of the image.

Yes	No
85.48%	14.51%

approximately 70% of the object categories improve their APRs. These results indicate that our approach learn better models from elements that are popular in the images of the training dataset. The overall improvement in the mAP is achieved mainly for the categories that are very common in both training and test images.

4.4.1. Cross-dataset evaluation

We tested the ability of our model to improve the object detection results across datasets. The model previously trained with the TVDT representations and the 2000 categories selected from the visual genome dataset were used to test images of the SUN2012 test dataset (Xiao et al., 2010). We have not used any annotations from the training or validation sets. Hence, we have not performed fine tunning or re-training of the networks. We also use the network weights of the Fast R-CNN learned from the visual genome. Our goal is to observe if the learned model generalizes to any image, instead of just learning a particular dataset.

The SUN2012 dataset has annotations for 3819 object categories, which are different from the 2.000 categories from our trained dataset. Only 950 categories from the 2000 categories were mapped to the existing ground truth categories of the SUN2012 dataset. The results for object detection using Fast R-CNN and our framework are reported in Table 3.

The performance of the Fast R-CNN model trained on the visual-genome dataset is 3.40 mAP, considering all the 950 available categories. The apparent low performance results in this dataset are due in part to the discrepancy between the categories that the model can predict and the available groundtruth for this dataset. The experiment shows that our frameworks allows to improve the results of object detection compared to the baseline Fast R-CNN. The resulting mAP is 4.12, which represents an improvement of 21.17% respect to Fast-RCNN object detection.

4.5. Obtained TDVT representation

The quality of the obtained TDVT representation is evaluated in the same way as we evaluate the visual representation obtained from parsing the dataset (Section 4.2). Again, we asked turkers to tell if the obtained TDVT visual tree properly represents the content of the image. A total of 119 participate in this experiment. The results are showed in Table 4.

According to the turkers, 85.48% of the images are represented correctly by the obtained TDVT output. These values are close to the results obtained for the parsing process, but the obtained TVDT representation was obtained automatically from images without annotations.

5. Conclusions

We proposed a new image representation that captures the structure of the image content, a methodology to extract the representation from existing object detection datasets, and algorithms for learning and inference of the proposed representation that allows to improve the results of object detection when a large number of categories are included. We performed our experiments in the challenging Visual Genome dataset obtaining large improvements, as measured by the mean Average Precision. The improvement is even kept on images from other dataset, as showed in the test set of the SUN2012 dataset. We showed that the proposed approach also allows to capture which ones are the most important objects, and obtain a model for the overall content of the image.

References

Amnih, A.M., Teh, Y.W., 2012. A fast and simple algorithm for training neural probabilistic language models. In: Proceedings of the 29th International Conference on Machine Learning.

Bell, S., Zitnick, C.L., Bala, K., Girshick, R., 2015. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. arXiv:1512.04143.

Choi, M.J., Lim, J.J., Torralba, A., Willsky, A.S., 2010. Exploiting hierarchical context on a large database of object categories. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR.

Dalal, N., Triggs, B., 2005. Histogram of oriented gradients for human detection. In: CVPR.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei -Fei, L., 2009. Imagenet: A large-scale hierarchical image database. In: CVPR.

Divvala, S.K., Hoiem, D., Hays, J.H., Efros, A.A., Hebert, M., 2009. An empirical study of context in object detection. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR.

Duan, L., Wu, C., Miao, J., Qing, L., Fu, Y., 2011. Visual saliency detection by spatially weighted dissimilarity. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR.

Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A., 2010. The PASCAL Visual Object Classes, VOC challenge. Int. J. Comput. Vis. 88 (2), 303–338.

Fang, H., Deng, L., Mitchell, M., Gupta, S., Dollar, P., Platt, J.C., Iandola, F., Gao, J., Zitnick, C.L., Srivastava, R.K., He, X., Zweig, G., 2015. From captions to visual concepts and back. In: CVPR.

Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D., 2010. Object detection with discriminatively trained part based models. IEEE Trans. Pattern Anal. Mach. Intell. 32 (9).

Fixot, R., 1957. Am. J. Ophthalmol..

Galleguillos, C., Belongie, S., 2008. Context based object categorization: A critical survey. In: ECCV.

Gidaris, S., Komodakis, N., 2015. Object detection via a multi-region & semantic segmentation-aware CNN model. arXiv:1505.01749 URL: http://arxiv.org/pdf/1505.01749v3.

Girshick, R., 2015. Fast R-CNN. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Deep residual learning for image recognition. arXiv:1512.03385.

Hendricks, L.A., Donahue, J., Donahue, J., Akata, Z., Rohrbach, M., Schiele, B., Darrell, T., 2016. Generating visual explanations. arXiv:1603.08507.

Jia, X., Gavves, E., Fernando, B., Tuytelaars, T., 2015. Guiding long-short term memory for image caption generation. In: ICCV.

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D.A., Bernstein, M., Fei -Fei, L., 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotation. arXiv:1602.07332.

Li, J., Wei, Y., Liang, X., Dong, J., Xu, T., Feng, J., Yan, S., 2016. Attentive contexts for object detection. arXiv:1603.07415.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., Zitnick, C.L., 2014a. Microsoft COCO: Common objects in context. In: ECCV.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C.Y., Berg, A.C., 2016. SSD: Single shot multibox detector. In: ECCV.

Mao, J., Huang, J., Toshev, A., Camburu, A., Yuille, A., Murphy, K., 2016. Generation and comprehension of unambiguous object descriptions. In: CVPR.

Miller, G.A., 1995. Wordnet: A lexical database for english. Commun. ACM 38 (11), 39–41.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection. In: CVPR.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, NIPS.

Sacks, O., 1989. Seeing voices. In: A Jouney into the World of Deaf. Vintage Books.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In: ICLR.

Song, Z., Chen, Q., Huang, Z., Hua, Y., Yan, S., 2011. Contextualizing object detection and classification. In: CVPR.

Thomee, B., Shamma, D.A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., Li, L.-J., 2016. YFCC100m: The new data in multimedia research. In: ACM Multimedia.

Uijlings, J., van de Sande, K., Gevers, T., Smeulders, A., 2013. Selective search for object recognition. IJCV 104 (2), 154–171.

Vaca-Castano, G., Das, S., Sousa, J.P., Lobo, N.D., Shah, M., 2016. Improved scene identification and object detection on egocentric vision of daily activities. Comput. Vis. Image Underst..

Vinyals, O., Toshev, A., Bengio, S., Erha, D., 2015. Show and tell: A neural image caption generator. In: CVPR.

Wu, Q., Shen, C., Liu, L., Dick, A., Dick, A., van den Hengel, A., 2015. What value do explicit high level concepts have in vision to language problems? arXiv:1506.01144.

Wu, Q., Wang, P., Shen, C., Dick, A., van den Henge, A., 2016. Ask me anything: free-form visual question answering based on knowledge from external sources. In: CVPR.

Xiao, J., Hays, J., Ehinger, K., Oliva, A., Torralba, A., 2010. Sun database: Large-scale scene recognition from abbey to zoo. In: CVPR.

You, Q., Jin, H., Wang, Z., Fang, C., Luo, J., 2016. Image captioning with semantic attention. In: CVPR.

Zhang, X., Lu, L., Lapata, M., 2016. Top-down tree long short-term memory networks. In: NAACL. URL: http://arxiv.org/abs/1511.00060.

Zhou, B., Tian, Y., Sukhbaatar, S., Szlam, A., Fergus, R., 2016. Simple baseline for visual question answering. 1512.02167v2.

Zitnick, C.L., Dollàr, P., 2014. Edge boxes: Locating object proposals from edges. In: European Conference in Computer Vision, ECCV.