

Iterative Visual Reasoning Beyond Convolutions

Xinlei Chen¹ Li-Jia Li² Li Fei-Fei² Abhinav Gupta¹
¹Carnegie Mellon University ²Google

Abstract

We present a novel framework for iterative visual reasoning. Our framework goes beyond current recognition systems that lack the capability to reason beyond stack of convolutions. The framework consists of two core modules: a local module that uses spatial memory [4] to store previous beliefs with parallel updates; and a global graph-reasoning module. Our graph module has three components: a) a knowledge graph where we represent classes as nodes and build edges to encode different types of semantic relationships between them; b) a region graph of the current image where regions in the image are nodes and spatial relationships between these regions are edges; c) an assignment graph that assigns regions to classes. Both the local module and the global module roll-out iteratively and cross-feed predictions to each other to refine estimates. The final predictions are made by combining the best of both modules with an attention mechanism. We show strong performance over plain ConvNets, e.g. achieving an 8.4% absolute improvement on ADE [55] measured by per-class average precision. Analysis also shows that the framework is resilient to missing regions for reasoning.

1. Introduction

In recent years, we have made significant advances in standard recognition tasks such as image classification [16], detection [37] or segmentation [3]. Most of these gains are a result of using feed-forward end-to-end learned ConvNet models. Unlike humans where visual reasoning about the space and semantics is crucial [1], our current visual systems lack any context reasoning beyond convolutions with large receptive fields. Therefore, a critical question is how do we incorporate both *spatial* and *semantic* reasoning as we build next-generation vision systems.

Our goal is to build a system that can not only extract and utilize hierarchy of convolutional features, but also improve its estimates via spatial and semantic relationships. But what are spatial and semantic relationships and how can they be used to improve recognition? Take a look at Fig. 1. An example of spatial reasoning (top-left) would be: if three regions out of four in a line are “window”, then the fourth is also likely to be “window”. An example of semantic reasoning (bottom-right) would be to recognize “school bus” even

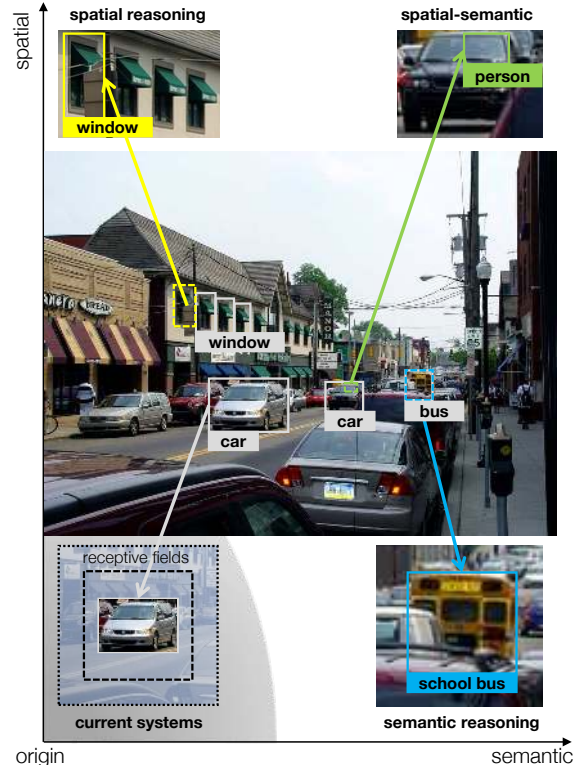


Figure 1. Current recognition systems lack the reasoning power beyond convolutions with large receptive fields, whereas humans can explore the rich space of spatial and semantic relationships for reasoning: e.g. inferring the fourth “window” even with occlusion, or the “person” who drives the “car”. To close this gap, we present a generic framework that also uses relationships to iteratively reason and build up estimates.

if we have seen few or no examples of it – just given examples of “bus” and knowing their connections. Finally, an example of spatial-semantic reasoning could be: recognition of a “car” on road should help in recognizing the “person” inside “driving” the “car”.

A key recipe to reasoning with relationships is to *iteratively* build up estimates. Recently, there have been efforts to incorporate such reasoning via top-down modules [38, 48] or using explicit memories [51, 32]. In the case of top-down modules, high-level features which have class-based information can be used in conjunction with low-level features to improve recognition performance. An

alternative architecture is to use explicit memory. For example, Chen & Gupta [4] performs sequential object detection, where a *spatial memory* is used to store previously detected objects, leveraging the power of ConvNets for extracting dense context patterns beneficial for follow-up detections.

However, there are two problems with these approaches: a) both approaches use stack of convolutions to perform *local* pixel-level reasoning [11], which can lack a *global* reasoning power that also allows regions farther away to directly communicate information; b) more importantly, both approaches assume enough examples of relationships in the training data – so that the model can learn them from scratch, but as the relationships grow exponentially with increasing number of classes, there is not always enough data. A lot of semantic reasoning requires learning from few or no examples [14]. Therefore, we need ways to exploit additional structured information for visual reasoning.

In this paper, we put forward a generic framework for both spatial and semantic reasoning. Different from current approaches that are just relying on convolutions, our framework can also learn from structured information in the form of knowledge bases [5, 56] for visual recognition. The core of our algorithm consists of two modules: the local module, based on spatial memory [4], performs pixel-level reasoning using ConvNets. We make major improvements on efficiency by parallel memory updates. Additionally, we introduce a global module for reasoning beyond local regions. In the global module, reasoning is based on a *graph* structure. It has three components: a) a knowledge graph where we represent classes as nodes and build edges to encode different types of semantic relationships; b) a region graph of the current image where regions in the image are nodes and spatial relationships between these regions are edges; c) an assignment graph that assigns regions to classes. Taking advantage of such a structure, we develop a reasoning module specifically designed to pass information on this graph. Both the local module and the global module roll-out iteratively and cross-feed predictions to each other in order to refine estimates. Note that, local and global reasoning are not isolated: a good image understanding is usually a compromise between background knowledge learned *a priori* and image-specific observations. Therefore, our full pipeline joins force of the two modules by an attention [3] mechanism allowing the model to rely on the most relevant features when making the final predictions.

We show strong performance over plain ConvNets using our framework. For example, we can achieve 8.4% absolute improvements on ADE [55] measured by per-class average precision, where by simply making the network deeper can only help $\sim 1\%$.

2. Related Work

Visual Knowledge Base. Whereas past five years in computer vision will probably be remembered as the successful resurgence of neural networks, acquiring visual knowledge at a large scale – the simplest form being labeled instances of objects [39, 30], scenes [55], relationships [25]

etc. – deserves at least half the credit, since ConvNets hinge on large datasets [44]. Apart from providing labels using crowd-sourcing, attempts have also been made to accumulate structured knowledge (*e.g.* relationships [5], *n*-grams [10]) automatically from the web. However, these works fixate on building knowledge bases rather than using knowledge for reasoning. Our framework, while being more general, is along the line of research that applies visual knowledge base to end tasks, such as affordances [56], image classification [32], or question answering [49].

Context Modeling. Modeling context, or the interplay between scenes, objects and parts is one of the central problems in computer vision. While various previous work (*e.g.* scene-level reasoning [46], attributes [13, 36], structured prediction [24, 9, 47], relationship graph [21, 31, 52]) has approached this problem from different angles, the breakthrough comes from the idea of feature learning with ConvNets [16]. On the surface, such models hardly use any explicit context module for reasoning, but it is generally accepted that ConvNets are extremely effective in aggregating local pixel-to-level context through its ever-growing receptive fields [54]. Even the most recent developments such as top-down module [50, 29, 43], pairwise module [40], iterative feedback [48, 34, 2], attention [53], and memory [51, 4] are motivated to leverage such power and depend on variants of convolutions for reasoning. Our work takes an important next step beyond those approaches in that it also incorporates learning from structured visual knowledge bases directly to reason with spatial and semantic relationships.

Relational Reasoning. The earliest form of reasoning in artificial intelligence dates back to symbolic approaches [33], where relations between abstract symbols are defined by the language of mathematics and logic, and reasoning takes place by deduction, abduction [18], *etc.* However, symbols need to be grounded [15] before such systems are practically useful. Modern approaches, such as path ranking algorithm [26], rely on statistical learning to extract useful patterns to perform relational reasoning on structured knowledge bases. As an active research area, there are recent works also applying neural networks to the graph structured data [42, 17, 27, 23, 35, 7, 32], or attempting to regularize the output of networks with relationships [8] and knowledge bases [20]. However, we believe for visual data, reasoning should be both local and global: discarding the two-dimensional image structure is neither efficient nor effective for tasks that involve regions.

3. Reasoning Framework

In this section we build up our reasoning framework. Besides plain predictions p_0 from a ConvNet, it consists of two core modules that reason to predict. The first one, local module, uses a spatial memory to store previous beliefs with parallel updates, and still falls within the regime of convolution based reasoning (Sec. 3.1). Beyond convolutions, we present our key contribution – a global module that reasons directly between regions and classes represented as nodes in a graph (Sec. 3.2). Both modules build up estimation it-

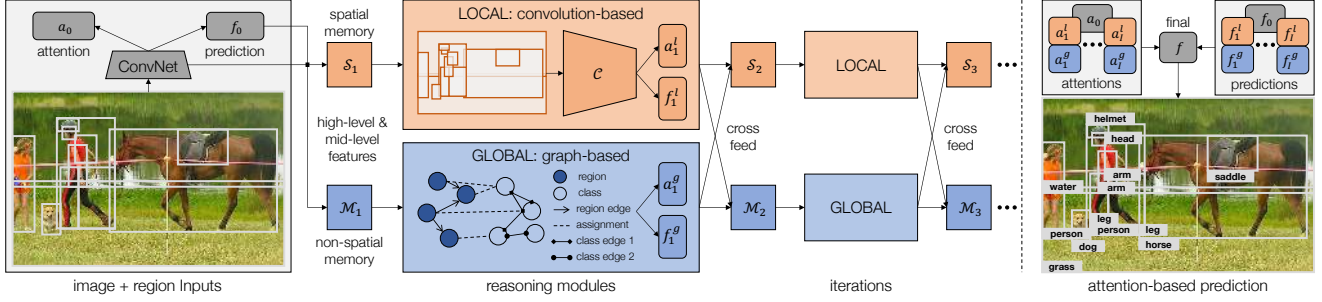


Figure 2. Overview of our reasoning framework. Besides a plain ConvNet that gives predictions, the framework has two modules to perform reasoning: a local one (Sec. 3.1) that uses spatial memory \mathcal{S}_i , and reasons with another ConvNet \mathcal{C} ; and a global one (Sec. 3.2) that treats regions and classes as nodes in a graph and reasons by passing information among them. Both modules receive combined high-level and mid-level features, and roll-out iteratively (Sec. 3.3) while cross-feeding beliefs. The final prediction f is produced by combining all the predictions f_i with attentions a_i (Sec. 3.4).

eratively (Sec. 3.3), with beliefs cross-fed to each other. Finally taking advantage of both local and global, we combine predictions from all iterations with an attention mechanism (Sec. 3.4) and train the model with sample re-weighting (Sec. 3.5) that focuses on hard examples (See Fig. 2).

3.1. Reasoning with Convolutions

Our first building block, the local module, is inspired from [4]. At a high level, the idea is to use a spatial memory \mathcal{S} to store previously detected objects at the very location they have been found. \mathcal{S} is a tensor with three dimensions. The first two, height H and width W , correspond to the reduced size ($1/16$) of the image. The third one, depth D ($=512$), makes each cell of the memory c a vector that stores potentially useful information at that location.

\mathcal{S} is updated with both high-level and mid-level features. For high-level, information regarding the estimated class label is stored. However, just knowing the class may not be ideal – more details about the shape, pose *etc.* can also be useful for other objects. For example, it would be nice to know the pose of a “person” playing tennis to recognize the “racket”. In this paper, we use the logits f before soft-max activation, in conjunction with feature maps from a bottom convolutional layer h to feed-in the memory.

Given an image region r to update, we first crop the corresponding features from the bottom layer, and resize it to a predefined square (7×7) with bi-linear interpolation as h_r . Since high-level feature f is a vector covering the entire region, we append it to all the 49 locations. Two 1×1 convolutions are used to fuse the information [4] and form our input features f_r for r . The same region in the memory \mathcal{S} is also cropped and resized to 7×7 , denoted as s_r . After this alignment, we use a convolutional gated recurrent unit (GRU) [6] to write the memory:

$$s'_r = u \circ s_r + (1 - u) \circ \sigma(W_f f_r + W_s(z \circ s_r) + b), \quad (1)$$

where s'_r is the updated memory for r , u is update gate, z is reset gate, W_f , W_s and b are convolutional weights and bias, and \circ is entry-wise product. $\sigma(\cdot)$ is an activation function. After the update, s'_r is placed back to \mathcal{S} with another

crop and resize operation¹.

Parallel Updates. Previous work [4] made sequential updates to memory. However, sequential inference is inefficient and GPU-intensive – limiting it to only give ten outputs per image [4]. In this paper we propose to update the regions in parallel as an approximation. In overlapping cases, a cell can be covered multiple times from different regions. When placing the regions back to \mathcal{S} , we also calculate a weight matrix Γ where each entry $\gamma_{r,c} \in [0, 1]$ keeps track of how much a region r has contributed to a memory cell c : 1 meaning the cell is fully covered by the region, 0 meaning not covered. The final values of the updated cell is the weighted average of all regions.

The actual reasoning module, a ConvNet \mathcal{C} of three 3×3 convolutions and two 4096-D fully-connected layers, takes \mathcal{S} as the input, and builds connections within the local window of its receptive fields to perform prediction. Since the two-dimensional image structure and the location information is preserved in \mathcal{S} , such an architecture is particularly useful for relationships with spatial reasoning.

3.2. Beyond Convolutions

Our second module goes beyond local regions and convolutions for global reasoning. Here the meaning of *global* is two-fold. First is *spatial*, that is, we want to let the regions farther away to directly communicate information with each other, not confined by the receptive fields of the reasoning module \mathcal{C} . Second is *semantic*, meaning we want to take advantage of visual knowledge bases, which can provide relationships between classes that are globally true (*i.e.* commonsense) across images. To achieve both types of reasoning, we build a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where \mathcal{N} and \mathcal{E} denote node sets and edge sets, respectively. Two types of nodes are defined in \mathcal{N} : region nodes \mathcal{N}_r for R regions, and class nodes \mathcal{N}_c for C classes.

As for \mathcal{E} , three groups of edges are defined between nodes. First for \mathcal{N}_r , a spatial graph is used to encode spatial relationships between regions ($\mathcal{E}_{r \rightarrow r}$). Multiple types

¹Different from previous work [4] that introduces an inverse operation to put the region back, we note that crop and resize *itself* with proper extrapolation can simply meet this requirement.

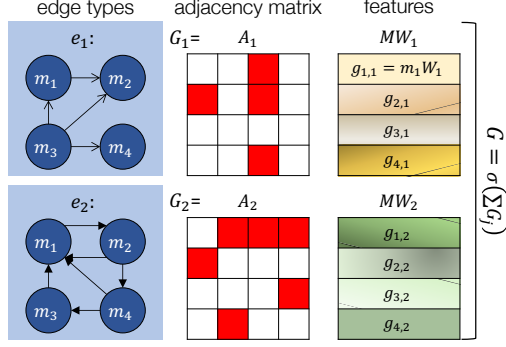


Figure 3. Illustration of directly passing information on a graph with multiple edge types. Here four nodes are linked with two edge types. Each node is represented as an input feature vector m_i (aggregated as M). Weight matrix W_j is learned for edge type j to transform inputs. Then adjacency matrix A_j is applied to pass information to linked nodes. Finally, output G is generated by accumulating all edge types and apply activation function.

of edges are designed to characterize the relative locations. We begin with basic relationships such as “left/right”, “top/bottom” and we define edge weights by measuring the pixel-level distances between the two. Note that we do not use the raw distance x directly, but instead normalizing it to $[0, 1]$ with a kernel $\kappa(x) = \exp(-x/\Delta)$ (where $\Delta=50$ is the bandwidth), with the intuition that closer regions are more correlated. The edge weights are then used directly in the adjacency matrix of the graph. Additionally, we include edges to encode the coverage patterns (e.g. intersection over union, IoU [12]), which can be especially helpful when two regions overlap.

A second group of edges lie between regions and classes, where the assignment for a region to a class takes place. Such edges shoulder the responsibility of propagating beliefs from region to class ($e_{r \rightarrow c}$) or backwards from class to region ($e_{c \rightarrow r}$). Rather than only linking to the most confident class, we choose full soft-max score p to define the edge weights of connections to all classes. The hope that it can deliver more information and thus is more robust to false assignments.

Semantic relationships from knowledge bases are used to construct the third group of edges between classes ($\mathcal{E}_{c \rightarrow c}$). Again, multiple types of edges can be included here. Classical examples are “is-kind-of” (e.g. between “cake” and “food”), “is-part-of” (e.g. between “wheel” and “car”), “similarity” (e.g. between “leopard” and “cheetah”), many of which are universally true and are thus regarded as commonsense knowledge for humans. Such commonsense can be either manually listed [39] or automatically collected [5]. Interestingly, even relationships beyond these (e.g. actions, prepositions) can help recognition [32]. Take “person ride bike” as an example, which is apparently more of an image-specific relationship. However, given less confident predictions of “person” and “bike”, knowing the relationship “ride” along with the spatial configurations of the two can also help prune other spurious explanations. To study both

cases, we experimented with two knowledge graphs in this paper: one created in-house with mostly commonsense edges, and the other also includes more types of relationships accumulated at a large-scale. For the actual graphs used in our experiments, please see Sec. 4.1 for more details.

Now we are ready to describe the graph-based reasoning module \mathcal{R} . As the input to our graph, we use $M_r \in \mathbb{R}^{R \times D}$ to denote the features from all the region nodes \mathcal{N}_r combined, where D ($=512$) is the number of feature channels. For each class node n_c , we choose off-the-shelf word vectors [22] as a convenient representation, denoted as $M_c \in \mathbb{R}^{C \times D}$. We then extend previous works [42, 35] and pass messages directly on \mathcal{G} (See Fig. 3). Note that, because our end-goal is to recognize regions better, all the class nodes should only be used as intermediate “hops” for better region representations. With this insight, we design two reasoning paths to learn the output features G_r : a *spatial* path on which only region nodes are involved:

$$G_r^{spatial} = \sum_{e \in \mathcal{E}_{r \rightarrow r}} A_e M_r W_e, \quad (2)$$

where $A_e \in \mathbb{R}^{r \times r}$ is the adjacency matrix of edge type e , $W_e \in \mathbb{R}^{d \times d}$ is weight (bias is ignored for simplicity). The second reasoning path is a *semantic* one through class nodes:

$$G_c^{semantic} = \sum_{e \in \mathcal{E}_{c \rightarrow c}} A_e \sigma(A_{e_{r \rightarrow c}} M_r W_{e_{r \rightarrow c}} + M_c W_c) W_e, \quad (3)$$

where we first map regions to classes through $A_{e_{r \rightarrow c}}$ and $W_{e_{r \rightarrow c}}$, combine the intermediate features with class features M_c , and again aggregate features from multiple types of edges between classes. Finally, the output for regions G_r are computed by merging these two paths:

$$G_r = \sigma(G_r^{spatial} + \sigma(A_{e_{c \rightarrow r}} G_c^{semantic} W_{e_{c \rightarrow r}})), \quad (4)$$

which first propagates semantic information back to regions, and then applies non-linear activation (See Fig. 4).

Just like convolution filters, the above-described paths can also be stacked, where the output G_r can go through another set of graph operations – allowing the framework to perform joint spatial-semantic reasoning with deeper features. We use three stacks of operations with residual connections [16] in \mathcal{R} , before the output is fed to predict.

3.3. Iterative Reasoning

A key ingredient of reasoning is to iteratively build up estimates. But how does information pass from one iteration to another? Our answer is *explicit* memory, which stores all the history from previous iterations. The local module uses spatial memory \mathcal{S} , and the global module uses another memory \mathcal{M} but without spatial structures. At iteration i , \mathcal{S}_i is followed by convolutional reasoning module \mathcal{C} to generate new predictions f_i^l for each region. Similarly, global

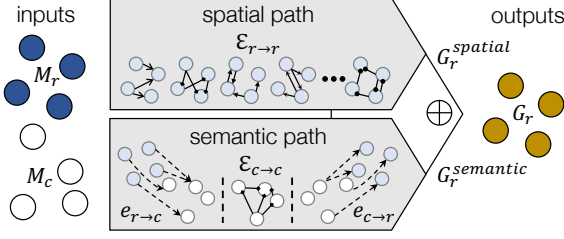


Figure 4. Two reasoning paths used in our global reasoning module \mathcal{R} . Taking the region and class inputs M_r and M_c , the spatial path directly passes information in the region graph with region-to-region edges $\mathcal{E}_{r \rightarrow r}$, whereas the semantic path first assigns regions to classes with $e_{r \rightarrow c}$, passes the information on to other classes with class-to-class edges $\mathcal{E}_{c \rightarrow c}$, and then propagates back. Final outputs are combined to generate output region features G_r .

module also gives new predictions f_i^g from \mathcal{R} . These new predictions as high-level features can then be used to get the updated memories \mathcal{S}_{i+1} and \mathcal{M}_{i+1} . The new memories will lead to another round of updated f_{i+1} s and the iteration goes on.

While one can do local and global reasoning in isolation, both the modules work best in conjunction. Therefore, for our full pipeline we want to join force of both modules when generating the predictions. To this end, we introduce *cross-feed* connections. After reasoning, both the local and global features are then concatenated together to update the memories \mathcal{S}_{i+1} and \mathcal{M}_{i+1} using GRU. In this way, spatial memory can benefit from global knowledge of spatial and semantic relationships, and graph can get a better sense of the local region layouts.

3.4. Attention

Inspired from the recent work on attention [3], we make another modification at the model output. Specifically, instead of only generating scores f , the model also has to produce an “attention” value a that denotes the relative confidence of the current prediction compared to the ones from other iterations or modules. Then the fused output is a weighted version of all predictions using attentions. Mathematically, if the model roll-outs I times, and outputs $N=2I+1$ (including I local, I global and 1 from plain ConvNet) predictions f_n , using attentions a_n , the final output f is calculated as:

$$f = \sum_n w_n f_n, \quad \text{where} \quad w_n = \frac{\exp(-a_n)}{\sum_{n'} \exp(-a_{n'})}. \quad (5)$$

Note again that here f_n is the logits before soft-max, which is then activated to produce p_n . The introduction of attention allows the model to intelligently choose feasible predictions from different modules and iterations.

3.5. Training

Finally, the overall framework is trained end-to-end, with a total loss function consists of: a) plain ConvNet loss \mathcal{L}_0 ; b) local module loss \mathcal{L}_i^l ; c) global module loss \mathcal{L}_i^g ; and d)

the final prediction loss with attentions \mathcal{L}_f .

Since we want our reasoning modules to focus more on the harder examples, we propose to simply *re-weight* the examples in the loss, based on predictions from previous iterations. Formally, for region r at iteration $i \geq 1$, the cross-entropy loss for both modules is computed as:

$$\mathcal{L}_i(r) = \frac{\max(1 - p_{i-1}(r), \beta)}{\sum_{r'} \max(1 - p_{i-1}(r'), \beta)} \log(p_i(r)), \quad (6)$$

where $p_i(r)$ is the soft-max output of the ground-truth class, and $\beta \in [0, 1]$ controls the entropy of the weight distribution: when $\beta=1$, it is uniform distribution; and when $\beta=0$, entropy is minimized. In our experiments, β is set to 0.5. $p_{i-1}(r)$ is used as features without back-propagation. For both local and global, $p_0(r)$ is the output from the plain ConvNet.

4. Experiments

In this section we evaluate the effectiveness of our framework. We begin with our experimental setups, which includes the datasets to work with (Sec. 4.1), the task to evaluate on (Sec. 4.2) and details of our implementation (Sec. 4.3). We discuss our results and analyze them in Sec. 4.4 and Sec. 4.5 respectively.

4.1. Datasets and Graphs

Datasets are biased [45]. For context reasoning we would naturally like to have scene-focused datasets [55] as opposed to object-focused ones [39]. To showcase the capabilities of our system, we need densely labeled dataset with a large number of classes. Finally, one benefit of using knowledge graph is to transfer across classes, therefore a dataset with *long-tail* distribution is an ideal test-bed. Satisfying all these constraints, ADE [55] and Visual Genome (VG) [25] where regions are densely labeled in open vocabulary are the main picks of our study.

For ADE, we use the publicly released training set (20,210) images for training, and split the validation set (2,000 images) into `val-1k` and `test-1k` with 1,000 images each. The original raw names are used due to a more detailed categorization [55]. We filter out classes with less than five instances, which leaves us with 1,484 classes. With the help of parts annotations in the dataset, a common-sense knowledge graph is created with five types of edges between classes: a) “is-part-of” (e.g. “leg” and “chair”); b) “is-kind-of” (e.g. “jacket” and “clothes”); c) “plural-form” (e.g. “tree” and “trees”); d) “horizontal-symmetry” (e.g. “left-arm” and “right-arm”); e) “similarity” (e.g. “handle” and “knob”). Notice that the first four types are directed edges, hence we also include their inverted versions.

For VG, the latest release (v1.4) is used. We split the entire set of 108,077 images into 100K, 4,077 and 4K as `train`, `val` and `test` set. Similar pre-processing is done on VG, except that we use synsets [39] instead of raw names due to less consistent labels from multiple annotators. 3,993 classes are used. For knowledge graph between

Table 1. Main results on ADE test-1k and VG test. AP is average precision, AC is classification accuracy. Superscripts show the improvement ∇ over the baseline.

%	Method	per-instance		per-class	
		AP [▽]	AC [▽]	AP [▽]	AC [▽]
ADE	Baseline	67.0	67.0	40.1	33.2
	w/ ResNet-101	68.2	68.3	40.8	34.4
	w/ 800-input	68.2	68.2	41.0	34.3
	Ensemble	68.7	68.8	42.9	35.3
	Ours-Local	71.6 ^{+4.6}	71.7 ^{+4.7}	47.9 ^{+7.8}	38.7 ^{+5.7}
	Ours-Global	69.8 ^{+2.8}	69.8 ^{+2.8}	44.5 ^{+4.4}	36.8 ^{+3.6}
	Ours-Final	72.6^{+5.6}	72.6^{+5.6}	48.5^{+8.4}	39.5^{+6.3}
VG	Baseline	49.1	49.6	16.9	12.1
	w/ ResNet-101	50.3	50.8	18.0	13.0
	w/ 800-input	49.5	50.0	17.0	12.2
	w/ Ensemble	50.2	50.7	17.7	12.3
	Ours-Local	51.4 ^{+2.3}	51.9 ^{+2.3}	18.8 ^{+1.9}	12.8 ^{+0.7}
	Ours-Global	50.9 ^{+1.8}	51.5 ^{+1.9}	18.3 ^{+1.4}	12.6 ^{+0.5}
	Ours-Final	51.7^{+2.6}	52.2^{+2.6}	19.1^{+2.2}	12.9 ^{+0.8}

classes, we take advantage of the relationship annotations in the set, and select the top 10 most frequent relationships to automatically construct edges beyond commonsense relationships constructed for ADE. For each type of relationships, the edge weights are normalized so that each row of the adjacency matrix is summed-up to one. While this approach results in a noisier graph, it also allows us to demonstrate that our approach is scalable and robust to noise.

Finally, we also show experiments on COCO [30]. However, since it is detection oriented – has only 80 classes picked to be mutually-exclusive, and covers less percentage of labeled pixels, we only report results a) without the knowledge graph and b) without a test split (trainval35k [4] for training and minival for evaluation). This setup is for analysis purposes only.

4.2. Task and Evaluation

We evaluate our system on the task of region classification, where the goal is to assign labels to designated regions denoted by rectangular bounding boxes. For both training and testing, we use provided ground-truth locations. We picked this task for three reasons. The **first** one is on evaluation. As the number of classes increases in the vocabulary, *missing* labels are inevitable, which is especially severe for object parts (e.g. “rim”, “arm”) and related classes (e.g. “shoes” vs. “sneakers”) where external knowledge is valuable. If there are missing labels, fair evaluation becomes much more difficult since accuracy becomes impossible to evaluate – cannot tell if a prediction is wrong, or the label itself is missing. Interestingly, such an issue also happens to other research areas (e.g. recommendation systems [41] and link prediction [28]). Borrowing ideas from them, a practical solution is to evaluate *only* on what we already know – in our case ground-truth regions. **Second**, although region classification is a simplified version of object detection and semantic segmentation, it maintains a richer set of la-

bels, especially including “stuff” classes like “road”, “sky”, and object instances. Modeling “stuff-object” and instance-level relationships is a crucial capability which would be missed in a pure detection/segmentation setting. **Finally** as our experiment will show (Sec. 4.5), while object detectors can be used off-the-shelf, the additional manually defined parameters and components (e.g. overlapping threshold for a region to be positive/negative, predefined scale/aspect ratio sets of anchors [37]) in its pipeline pose limitations on how much context can benefit. For example, after non-maximal suppression (NMS), highly overlapping objects (e.g. “window” and “shutter”) will be suppressed, and ironically this is exactly where context reasoning could have helped. On the other hand, by feeding fixed regions directly for end-to-end learning, we can at least factorize the *recognition* error from the *localization* one [19], and get a clean focus on how context can help discriminating confusing classes.

Since ADE is a segmentation dataset, we convert segmentation masks to bounding boxes. For object classes (e.g. “person”), each instance is created a separate box. Part (e.g. “head”) and part-of-part (e.g. “nose”) are also included. For VG and COCO, boxes are directly used.

For evaluation, we use classification accuracy (AC) and average precision (AP) [12]. Note that since all the regions are fixed with known labels, there is no need to set a region overlap threshold for AP. Results can be aggregated in two ways: the first way (“per-class”) computes metrics separately for each class in the set, and take the mean; since the final scores are all taken from a calibrated soft-max output, a second way (“per-instance”) that computes metrics simultaneously for all classes. Intuitively, “per-class” assigns more weights to instances from rare classes.

4.3. Implementation Details

A simplified version of *tf-faster-rcnn*² is used to implement our baseline for region classification, with region proposal branch and bounding box regression components removed. Unless otherwise noted, ResNet-50 [16] pre-trained on ImageNet [39] is used as our backbone image classifier, and images are enlarged to shorter size 600 pixels during both training and testing. Specifically, full-image shared convolutional feature maps are computed till the last *conv4* layer. Then the ground-truth boxes are used as regions-of-interest to compute region-specific features (crop and resize to 7×7 without max-pool). All layers of *conv5* and up are then adopted to obtain the final feature for the baseline prediction p_0 . Batch normalization parameters are fixed.

For the local module, we use the last *conv4* layer as our mid-level features to feed the spatial memory \mathcal{S} . For the global module, mid-level features are the final *conv5* (2048-D) layer after avg-pool. Both features are fused with the logits before soft-max f , and then fed into the memory cells. Word vectors from fastText [22] are used to represent each class, which extracts sub-word information and

²<https://github.com/endernewton/tf-faster-rcnn>

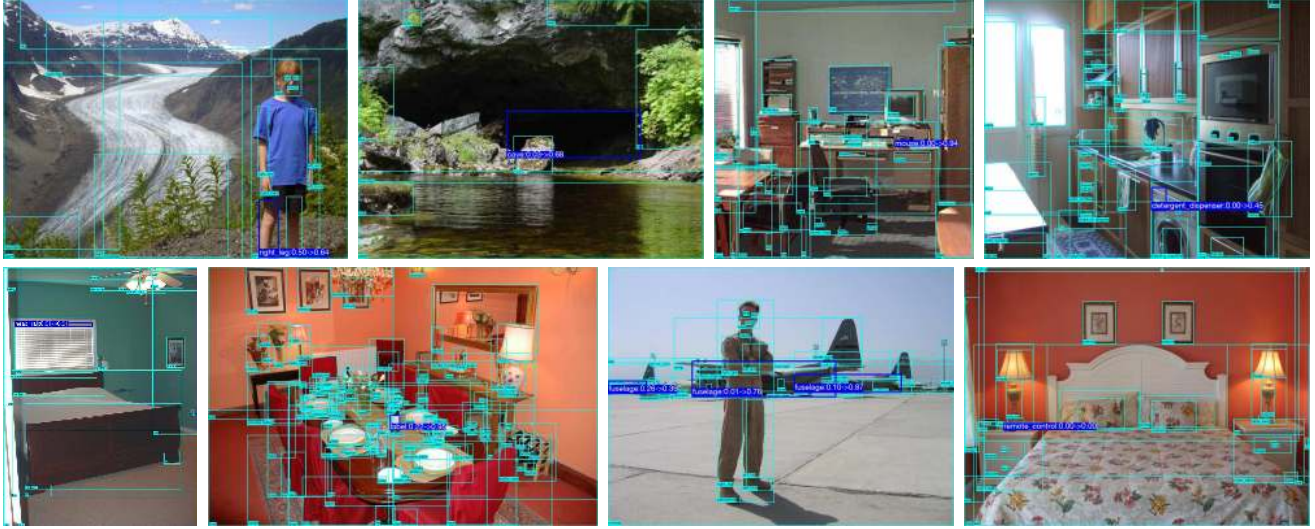


Figure 5. Qualitative examples from ADE test-1k (best if zoomed-in). For regions highlighted in blue, the predictions from baseline and our model are compared. Other regions are also listed to provide the context. For example, the “right-leg” is less confused with “left-leg” after reasoning (top-left); the “mouse” on the “desk” is predicted despite low resolution (top-third); and “detergent-dispenser” is recognized given the context of “washing-machine” (top-right). At bottom-right we show a failure case where context does not help “remote-control”, probably because it has never appeared on the “night-table” before and no semantic relationship is there to help.

generalizes well to out-of-vocabulary words. ReLU is selected as the activation function. We roll-out the reasoning modules 3 times and concurrently update all regions at each iteration, as more iterations do not offer more help.

We apply stochastic gradient descent with momentum to optimize all the models, and use the validation set to tune hyper-parameters. Our final setups are: $5e^{-4}$ as the initial learning rate, reduced once ($0.1\times$) during fine-tuning; $1e^{-4}$ as weight decay; 0.9 as momentum. For ADE, we train 320K iterations and reduce learning rate at 280K. For VG and COCO the numbers are 640K/500K and 560K/320K, respectively³. We use a single image per step, and the only data augmentation technique used during training is left-right flipping⁴. No augmentation is used in testing.

4.4. Main Results

Quantitative results on ADE test-1k and VG test are shown in Tab. 1. Besides plain ConvNet p_0 , we also add three more baselines. First, we use ResNet-101 as the backbone to see the performance can benefit from deeper networks. Second, we increase the input image size with a shorter side 800 pixels, which is shown helpful especially for small objects in context [29]. Finally, to check whether our performance gain is a result of more parameters, we include model ensemble as the third baseline where the prediction of two separate baseline models are averaged.

As can be seen, our reasoning modules are performing much better than all the baselines on ADE. The local module alone can increase per-class AP by 7.8 absolute points. Although the global module alone is not as effective (4.4%

improvement), the performance gain it offers is *complementary* to the local module, and combining both modules we arrive at an AP of 48.5% compared to the baseline AP 40.1%. On the other hand, deeper network and larger input size can only help $\sim 1\%$, less than model ensembles. Additionally, our models achieve higher per-class metric gains than per-instance ones, indicating that *rare* classes get helped more – a nice property for learning from few examples. Some qualitative results are listed in Fig. 5.

We also report the speed for future reference. On Titan Xp, the final model on ADE trains at 0.344s per iteration, compared to the baseline ResNet-50 at 0.163s and ResNet-101 at 0.209s. For testing, our model takes 0.165s, whereas ResNet-50 0.136s, ResNet-101 0.156s. We believe the additional cost is minimal with regard to the extra accuracy.

We see a similar but less significant trend on VG. This can potentially be a result of *noisier* labels – for ADE (and COCO shown later), the per-instance AP and AC values are within 0.1%, intuitively suggesting that *higher* scores usually correspond to correct classifications. However, on VG the difference is at $\sim 0.5\%$, meaning more of the highly confident predictions are not classified right, which are likely caused by missing ground-truths.

4.5. Analysis

Our analysis is divided into two major parts. In the first part, we conduct thorough ablative analysis on the framework we have built. Due to space limitation, we only report results on ADE here at Tab. 2, for more analysis on VG, please check our supplementary material.

As can be seen, re-weighting hard examples with Eq. 6 helps around 0.5% regardless of reasoning modules. Spatial memory \mathcal{S} is critical in the local module – if replaced

³Training longer still reduces cross-entropy, but drops both AP and AC.

⁴The labels for class pairs like “left-hand” and “right-hand” are swapped for flipped images.

Table 2. Ablative analysis on ADE test-1k. In the first row of each block we repeat Local, Global and Final results from Tab. 1. Others see Sec. 4.5 for details.

%	Analysis	per-instance		per-class	
		AP	AC	AP	AC
Local	Ours _{.Local}	71.6	71.7	47.9	38.7
	w/o re-weight	71.3	71.3	46.7	37.9
	w/o \mathcal{C}	70.9	71.0	46.1	37.5
	w/o \mathcal{S}	67.6	67.6	42.1	34.4
Global	Ours _{.Global}	69.8	69.8	44.5	36.8
	w/o re-weight	69.2	69.2	43.8	36.7
	w/o spatial	67.8	67.8	41.5	35.0
	w/o semantic	69.1	69.2	43.9	35.9
	w/o \mathcal{R}	67.1	67.2	41.5	34.5
	w/o \mathcal{M} & \mathcal{R}	67.1	67.1	41.0	34.0
Final	Ours _{.Final}	72.6	72.6	48.5	39.5
	w/o re-weight	72.1	72.2	47.3	38.6
	w/o cross-feed	72.2	72.2	47.6	39.0
	2 iterations	71.9	72.0	48.1	39.0

by feeding last *conv4* layer directly the performance drops almost to baseline. Local context aggregator \mathcal{C} is less influential for ADE since the regions including background are densely labeled. A different story takes place at the global module: removing the reasoning module \mathcal{R} steeply drops performance, whereas further removing memory \mathcal{M} does not hurt much. Finally, for our full pipeline, removing cross-feeding and dropping the number of iterations both result in worse performance.

Missing Regions. So far we have shown results when all the regions are present. Next, we want to analyze if our framework is robust to missing regions: if some percentage of regions are not used for reasoning. This will be a common scenario if we use our framework in the detection setting – the underlying region proposal network [37] may itself miss some regions. We perform this set of experiments on COCO, since its regions are object-focused.

We test three variations. In the first variation, the same region classification pipeline is applied as-is. In the other two, we drop regions. While we could have done it randomly, we simulate the real-world scenario by using region proposals from faster R-CNN [37] (1190K/900K, minival detection mAP 32.4%) for testing, where 300 region proposals after NMS are applied to filter the ground-truth regions (max IoU $> \delta$). Evaluation is only done on the remaining regions. Here we choose not to use region proposals directly, since the model has seen ground truth regions only. We test two variations: a) “pre”, where the regions are filtered before inference, *i.e.* only the remaining ground-truths are fed for reasoning; “post”, where regions are filtered after inference. Note that for the baseline, “pre” and “post” makes no difference performance-wise.

The results are summarized in Tab. 3. Interestingly, despite lacking a knowledge graph, our global module works better than the local module, likely due to its power

Table 3. Results with missing regions when region proposals are used. COCO minival is used since it is more detection oriented. **pre** filters regions before inference, and **post** filters after inference.

Method	pre	post	per-instance		per-class	
			AP $^{\nabla}$	AC $^{\nabla}$	AP $^{\nabla}$	AC $^{\nabla}$
Baseline			83.2	83.2	83.7	75.9
Ours _{.Local}			84.9 ^{+1.7}	84.9 ^{+1.7}	85.8 ^{+2.1}	77.6 ^{+1.7}
Ours _{.Global}			85.6 ^{+2.4}	85.7 ^{+2.5}	86.9 ^{+3.2}	78.2 ^{+2.3}
Ours _{.Final}			86.0^{+2.8}	86.0^{+2.8}	87.4^{+3.7}	79.0^{+3.1}
Baseline	-	-	87.0	87.0	87.7	80.2
Ours _{.Final}	✓		88.6 ^{+1.6}	88.6 ^{+1.6}	89.9 ^{+2.2}	82.6^{+2.4}
Ours _{.Final}		✓	88.8^{+1.8}	88.8^{+1.8}	90.1^{+2.4}	82.5 ^{+2.3}

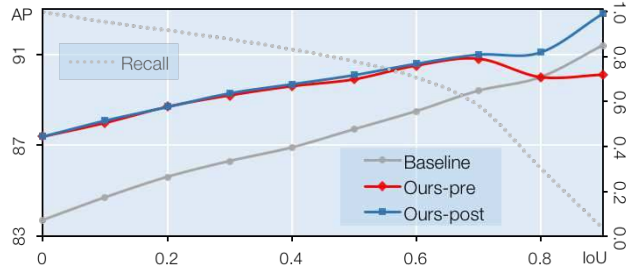


Figure 6. Trends of recall and per-class AP when varying IoU threshold δ from 0 to .9 to drop regions. See text for details.

that allows direct region-to-region communication even for farther-away pairs. Combining the two, we report 3.7% absolute advantage on per-class AP over the baseline even with all classes being objects – no “stuff” classes involved.

In Fig. 6, we vary δ from 0 to .9: with 0 keeping all regions and 0.9 dropping the most. As the trend shows, while the reasoning module suffers when regions are dropped, it is quiet resilient and the performance degradation is smooth. For example (listed in Tab. 3), with an IoU threshold δ of 0.5 that recalls 78.1% of the ground truth boxes, we still outperform the baseline by 2.4% in the “post” setting, and 2.2% in “pre” where not all regions can be fed for reasoning. The lower gap implies a) region proposals are usually corresponding to easy examples where less context is needed, and b) context reasoning frameworks like ours benefit from more known regions. At $\delta=.8$ the recall (30.5%) is so small that it cannot afford much reasoning, and at $\delta=.9$ (recall 3.9%), reasoning even hurts the performance.

5. Conclusion

We presented a novel framework for iterative visual reasoning. Beyond convolutions, it uses a graph to encode spatial and semantic relationships between regions and classes and passes message on the graph. We show strong performance over plain ConvNets, *e.g.* achieving an 8.4% absolute gain on ADE and 3.7% on COCO. Analysis also shows that our reasoning framework is resilient to missing regions caused by current region proposal approaches.

Acknowledgements: This work was supported in part by ONR MURI N000141612007. XC would also like to thank Shengyang Dai and Google Cloud AI team for support during the internship.

References

- [1] I. Biederman, R. J. Mezzanotte, and J. C. Rabinowitz. Scene perception: Detecting and judging objects undergoing relational violations. *Cognitive psychology*, 14(2):143–177, 1982. 1
- [2] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. In *CVPR*, 2016. 2
- [3] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016. 1, 2, 5
- [4] X. Chen and A. Gupta. Spatial memory for context reasoning in object detection. *arXiv preprint arXiv:1704.04224*, 2017. 1, 2, 3, 6
- [5] X. Chen, A. Shrivastava, and A. Gupta. Neil: Extracting visual knowledge from web data. In *ICCV*, 2013. 2, 4
- [6] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*, 2014. 3
- [7] R. Das, A. Neelakantan, D. Belanger, and A. McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426*, 2016. 2
- [8] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-scale object classification using label relation graphs. In *ECCV*, 2014. 2
- [9] C. Desai, D. Ramanan, and C. C. Fowlkes. Discriminative models for multi-class object layout. *IJCV*, 95(1):1–12, 2011. 2
- [10] S. K. Divvala, A. Farhadi, and C. Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*, 2014. 2
- [11] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert. An empirical study of context in object detection. In *CVPR*, 2009. 2
- [12] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 4, 6
- [13] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 2
- [14] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *TPAMI*, 28(4):594–611, 2006. 2
- [15] S. Harnad. The symbol grounding problem. *Physica D: Non-linear Phenomena*, 42(1-3):335–346, 1990. 2
- [16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 4, 6
- [17] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 2
- [18] J. R. Hobbs, M. Stickel, P. Martin, and D. Edwards. Interpretation as abduction. In *ACL*, 1988. 2
- [19] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. *ECCV*, 2012. 6
- [20] Z. Hu, Z. Yang, R. Salakhutdinov, and E. P. Xing. Deep neural networks with massive learned knowledge. In *EMNLP*, 2016. 2
- [21] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *CVPR*, 2015. 2
- [22] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016. 4, 6
- [23] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 2
- [24] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011. 2
- [25] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. Visual genome: Connecting language and vision using crowd-sourced dense image annotations. *arXiv:1602.07332*, 2016. 2, 5
- [26] N. Lao, T. Mitchell, and W. W. Cohen. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, 2011. 2
- [27] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015. 2
- [28] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the Association for Information Science and Technology*, 58(7):1019–1031, 2007. 6
- [29] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. *arXiv:1612.03144*, 2016. 2, 7
- [30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2, 6
- [31] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei. Visual relationship detection with language priors. In *ECCV*, 2016. 2
- [32] K. Marino, R. Salakhutdinov, and A. Gupta. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844*, 2016. 1, 2, 4
- [33] A. Newell. Physical symbol systems. *Cognitive science*, 4(2):135–183, 1980. 2
- [34] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 2
- [35] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, 2016. 2, 4
- [36] D. Parikh and K. Grauman. Relative attributes. In *ICCV*, 2011. 2
- [37] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv:1506.01497*, 2015. 1, 6, 8
- [38] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1
- [39] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 2, 4, 5, 6
- [40] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*, 2017. 2
- [41] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001. 6

- [42] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *TNN*, 20(1):61–80, 2009. [2](#), [4](#)
- [43] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond Skip Connections: Top-Down Modulation for Object Detection. *arXiv:1612.06851*, 2016. [2](#)
- [44] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *ICCV*, 2017. [2](#)
- [45] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011. [5](#)
- [46] A. Torralba, K. P. Murphy, W. T. Freeman, M. A. Rubin, et al. Context-based vision system for place and object recognition. In *ICCV*, 2003. [2](#)
- [47] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *TPAMI*, 32(10):1744–1757, 2010. [2](#)
- [48] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, 2016. [1](#), [2](#)
- [49] Q. Wu, P. Wang, C. Shen, A. Dick, and A. van den Hengel. Ask me anything: Free-form visual question answering based on knowledge from external sources. In *CVPR*, 2016. [2](#)
- [50] S. Xie, X. Huang, and Z. Tu. Top-down learning for structured labeling with convolutional pseudoprior. In *ECCV*, 2016. [2](#)
- [51] C. Xiong, S. Merity, and R. Socher. Dynamic memory networks for visual and textual question answering. *arXiv*, 1603, 2016. [1](#), [2](#)
- [52] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei. Scene graph generation by iterative message passing. *arXiv preprint arXiv:1701.02426*, 2017. [2](#)
- [53] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola. Stacked attention networks for image question answering. In *CVPR*, 2016. [2](#)
- [54] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. [2](#)
- [55] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ade20k dataset. *arXiv preprint arXiv:1608.05442*, 2016. [1](#), [2](#), [5](#)
- [56] Y. Zhu, C. Zhang, C. Ré, and L. Fei-Fei. Building a large-scale multimodal knowledge base system for answering visual queries. *arXiv:1507.05670*, 2015. [2](#)