Spatial Memory for Context Reasoning in Object Detection

Xinlei Chen Abhinav Gupta
School of Computer Science, Carnegie Mellon University
{xinleic,abhinavg}@cs.cmu.edu

Abstract

Modeling instance-level context and object-object relationships is extremely challenging. It requires reasoning about bounding boxes of different classes, locations etc. *Above all, instance-level spatial reasoning inherently requires modeling conditional distributions on previous detections. Unfortunately, our current object detection systems do not have any* **memory** *to remember what to condition on! The state-of-the-art object detectors still detect all object in parallel followed by non-maximal suppression (NMS). While memory has been used for tasks such as captioning, they mostly use image-level memory cells without capturing the spatial layout. On the other hand, modeling object-object relationships requires* **spatial** *reasoning – not only do we need a memory to store the spatial layout, but also a effective reasoning module to extract spatial patterns. This paper presents a conceptually simple yet powerful solution – Spatial Memory Network (SMN), to model the instance-level context efficiently and effectively. Our spatial memory essentially assembles object instances back into a pseudo "image" representation that is easy to be fed into another ConvNet for object-object context reasoning. This leads to a new sequential reasoning architecture where image and memory are processed in parallel to obtain detections which update the memory again. We show our SMN direction is promising as it provides 2.2% improvement over baseline Faster RCNN on the COCO dataset with VGG16*[1].

1. Introduction

Context helps image understanding! Apart from strong psychological evidence [4, 37, 64, 65] that context is vital for humans to recognize objects, many empirical studies in the computer vision community [10, 11, 16, 22, 23, 43, 60, 62, 73, 82, 83] have also suggested that recognition algorithms can be improved by proper modeling of context.

But what is the right model for context? Consider the problem of object detection. There are two common models of context often used in the community. The first type of model incorporates image or scene level context [5, 36, 46, 58, 63, 74, 81]. The second type models object-object relationships at instance-level [15, 27, 32, 58, 67, 94]. Take the the first image of Fig. 1 as an example, both the *person* and the *tennis racket* can be used to create a contextual prior on where the *ball* should be.

[1]For more up-to-date results, please check our arXiv submission https://arxiv.org/abs/1704.04224.

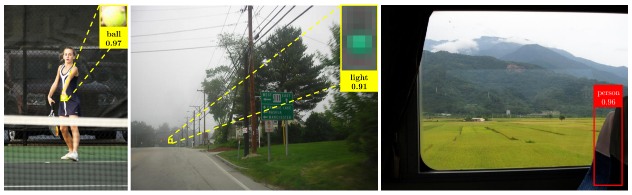

Figure 1: Evidence of image-level context reasoning inside ConvNets. All examples are from our baseline faster RCNN detector with VGG16 `conv5_3` features on COCO [52]. Numbers are class confidences. Left and middle: two examples where the ConvNet is able to detect tiny and simple-shaped objects much smaller than the receptive field size. Right: a false positive detection for person given the seat on a passenger train. Our Spatial Memory Network takes advantage of this power by encoding multiple object instances into a "pseudo" image representation.

Of these two models, which one is more effective for modeling context? A quick glimpse on the current state-of-the-art approaches, the idea of single region classification [40, 49, 51, 53, 69] with deep ConvNets [33, 76] is still dominating object detection. On the surface, these approaches hardly use any contextual reasoning; but we believe the large receptive fields of the neurons in fact do incorporate image-level context (See Fig. 1 for evidences). On the other hand, there has been little or no success in modeling object-object relationships or instance-level context in recent years.

Why so? Arguably, modeling the instance-level context is more challenging. Instance-level reasoning for object detection would have to tackle bounding boxes pairs or groups in different classes, locations, scales, aspect ratios, *etc.* Moreover, for modeling image-level context, the grid structure of pixels allows the number of contextual inputs to be reduced efficiently (*e.g.* to a local neighborhood [11, 43, 73] or a smaller scale [76, 87]), whereas such reductions for arbitrary instances appear to be not so trivial. Above all, instance-level spatial reasoning inherently requires modeling *conditional* distributions on previous detections, but our current object detection systems do not have any **memory** to remember what to condition on! Even in the case of multi-class object detection, the joint layout [15] is estimated by detecting all objects in parallel followed by non-maximal suppression (NMS) [20]. What we need is an object detection system with memory built inside it!

Memory has been successfully used in the recognition community recently for tasks such as captioning [13, 17, 59, 84, 85, 92] or visual question answering [2, 3, 24, 44, 55, 57, 72, 90, 91, 93, 97]. However, these works mostly focus on modeling an image-level memory, without capturing the spatial layout of the understanding so far. On the other hand, modeling object-object relationships requires **spatial** reasoning – not only do we need a memory to store the spatial layout, but also a suitable reasoning module to extract spatial patterns. This paper presents a conceptually simple yet powerful solution – Spatial Memory Network (SMN), to model the instance-level context efficiently and effectively. Our key insight is that the best spatial reasoning module is a ConvNet itself! In fact, we argue that ConvNets are actually the most generic[2] and effective framework for extracting spatial and contextual information so far! Inspired by this observation, our spatial memory essentially assembles object instances back into a pseudo "image" representation that is easy to be fed into another ConvNet to perform object-object context reasoning.

However, if ConvNets are already so excellent at modeling context, why would we even bother something else? Isn't the image itself the ultimate source of information and therefore the best form of "spatial memory"? Given an image, shouldn't an ultra-deep network already take care of the full reasoning inside its architecture? In spite of these valid concerns, we argue that a spatial memory still presents as an important next step for object detection and other related tasks, for the following reasons:

- First, we note that current region-based object detection methods are still treating object detection as a *perception* problem, not a *reasoning* problem: the region classifier still produces multiple detection results around an object instance during inference, and relies on manually designed NMS [69] with a pre-defined threshold for de-duplication. This process can be sub-optimal. We show that with a spatial memory that memorizes the already detected objects, it is possible to learn the functionality of NMS automatically.

- Second, replacing NMS is merely a first demonstration for context-based reasoning for object detection. Since the spatial memory is supposed to store both semantic and location information, a legitimate next step would be full context reasoning: *i.e.*, infer the "what" and "where" of other instances based on the current layout of detected objects in the scene. We show evidence for such benefits on COCO [52].

- Third, our spatial memory essentially presents as a general framework to encode instance-level visual knowledge [48], which requires the model to properly handle the spatial (*e.g.* overlaps) and semantic (*e.g.* poses) interactions between groups of objects. Our approach follows the spirit of end-to-end learning, optimizing the representation for an end-task – object detection. Both the representation and the idea can be applied to other tasks that require holistic image understanding [3, 42, 98].

[2]Many context models can be built or formulated as ConvNets [89, 96].

2. Related Work

As we already mentioned most related work for context and memory in Sec. 1, in this section we mainly review ideas that use sequential prediction for object detection. A large portion of the literature [28, 45, 56] focuses on sequential approaches for region proposals (*i.e.*, foreground/background classification). The motivation is to relieve the burden for region classifiers by replacing an exhaustive sliding-window search [20] with a smarter and faster search process. In the era of ConvNet-based detectors, such methods usually struggle to keep a delicate balance between efficiency and accuracy, since a convolution based 0/1 classifier (*e.g.* region proposal network [69]) already achieves an impressive performance when maintaining a reasonable speed. Sequential search has also been used for localizing small landmarks [77], but the per-class model assumes the existence of such objects in an image and lacks the ability to use other categories as context.

Another commonly used trick especially beneficial for reducing localization error is iterative bounding box refinement [25, 26, 69, 95], which leverages local image context to predict a better bounding box iteratively. This line of research is complementary to our SMN, since its goal is to locate the original instance *itself* better, whereas our focus is on how to better detect *other* objects given the current detections.

An interesting recent direction focuses on using deep reinforcement learning (DRL) to optimize the sequence selection problem in detection [6, 9, 50, 61]. However, due to the lack of full supervision signal in a problem with high-dimensional action space[3], DRL has so far only been used for bounding box refinements or knowledge-assisted detecton, where the action space is greatly reduced. Nevertheless, SMN can naturally serve as an encoder of the state in a DRL system to directly optimize average precision [34].

Note that the idea of using higher-dimensional memory in vision is not entirely new. It has resemblance to spatial attention, which has been explored in many high-level

[3]Jointly reason about all bounding boxes and all classes.

tasks [47, 68, 91, 92]. To bypass NMS, LSTM [35] cells arranged in 2D order [78] and intersection-over-union (IoU) maps [39] have been used for single-class object detection. We also notice a recent trend in using 2D memory as a map for planning and navigation [31, 66]. Our work extends such efforts into generic, multi-class object detection, performing joint reasoning on both space and semantics.

3. Background: Faster RCNN

Our spatial memory network is agnostic to the choice of base object detection model. In this paper we build SMN on top of Faster R-CNN [69] (FRCNN) as a demonstration, which is a state-of-the-art detector that predicts and classifies Regions of Interest (RoIs). Here we first give a brief review of the approach.

3.1. Base Network

We use VGG16 [76] as the base network for feature extraction. It has 13 convolutional (conv), 5 max-pooling (pool), and 2 fully connected (fc) layers before feeding into the final classifier, and was pre-trained on the ILSVRC challenge [71]. Given an image $\mathcal{I}$ of height h and width w, feature maps from the last conv layer (conv5_3) are first extracted by FRCNN. The conv5_3 feature size (h', w') is roughly γ=1/16 of the original image in each spatial dimension. On top of it, FRCNN proceeds by allocating two sub-networks for region proposal and region classification.

3.2. Region Proposal

The region proposal network essentially trains a class-agnostic objectness [1] classifier, proposing regions that are likely to have a foreground object in a sliding window manner [20]. It consists of 3 conv layers, one maps from conv5_3 to a suitable representation for RoI proposals, and two 1×1 siblings on top of this representation for foreground/background classification and bounding box regression. Note that at each location, anchor boxes [69] of multiple scales (s) and aspect ratios (r) are used to cover a dense sampling of possible windows. Therefore the total number of proposed boxes is $K{\approx}h'\times w'\times s\times r$[4]. During training and testing, $k{\ll}K$ regions are selected by this network as candidates for the second-stage region classification.

3.3. Region Classification

Since the base network is originally an image classifier, region classification network inherits most usable parts of VGG16, with two caveats. First, because RoI proposals can be be arbitrary rectangular bounding boxes, RoI pooling [26, 40] is used in place of pool on conv5_3 to match the the square-sized (7×7) input requirement for fc6. Second, the 1,000-way fc layer for ILSVRC classification is replaced by two fc layers for C-way classification and bounding box regression respectively. Each of the C classes gets a separate bounding box regressor.

[4]Boarder anchors excluded.

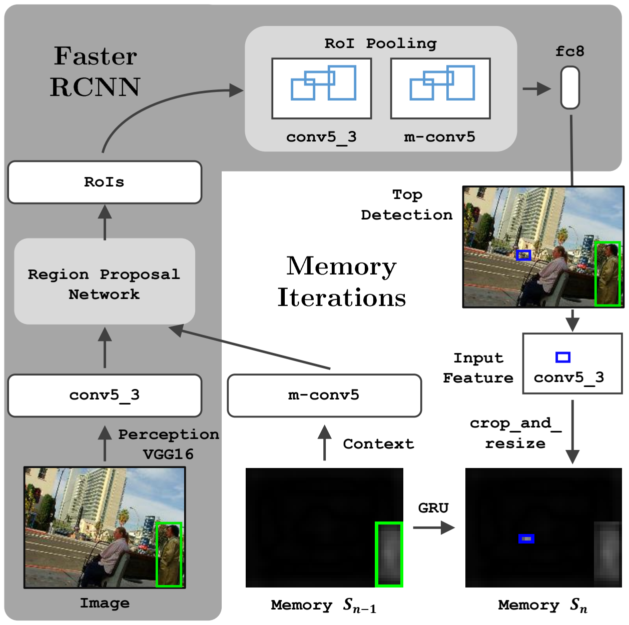

Figure 2: Overview of memory iterations for object detection. The original components from FRCNN are shown in the gray area. The old detection (*person*) is marked with a green box, and the new detection (*car*) is marked with blue. Here the network is unrolled one iteration.

3.4. De-duplication

We want to point out the often-neglected fact that a standard post-processing step is used in almost all detectors [20, 49, 53, 69] to disambiguate duplications – NMS. For FRCNN, NMS takes place in both stages. First, for region proposals, it prunes out the overlapping RoIs that are likely corresponding to the same object ("one-for-all-class") to train the region classifier. Second, for the final detection results, NMS is applied in an isolated, per-class manner ("one-for-each-class"). In this paper, we still use NMS for RoI sampling during training [12], and mainly focus on building a model to replace the per-class NMS, with the hope that the model can encode the rich interplay across multiple classes when suppressing redundant detections.

4. Spatial Memory Network

To better motivate the use of spatial memory network, we resort to a mathematical formulation of the task at hand. For object detection, the goal is to jointly infer and detect all the object instances $\mathcal{O}{=}[O_1, O_2, O_3, \cdots, O_N]$ given an image $\mathcal{I}$, where N is the maximum number of object instances for any image[5]. Then the objective function of training a model

[5]O_n denotes both the class and location of the object instance. When there is not enough foreground objects, the sequence can be padded with the background class.

(*e.g*. FRCNN) $\mathcal{M}$ is to maximize the log-likelihood:

$$\begin{aligned}\arg\max_{\mathcal{M}} \mathcal{L} &= \log \mathbb{P}(O_{1:N}|\mathcal{M}, \mathcal{I}) \\ &= \sum_{n=1:N} \log \mathbb{P}\left(O_n|\mathcal{O}_{0:n-1}, \mathcal{M}, \mathcal{I}\right), \end{aligned} \quad (1)$$

where $\mathcal{O}_{0:n-1}$ is short for $[O_1, O_2, O_3, \cdots, O_{n-1}]$ and $\mathcal{O}_{0:0}$ is an empty set. Note that this decomposition of the joint layout probability is exact [18], regardless of the order we are choosing.

For a region-based object detector, Eq.(1) is approximated by detecting each object instance separately:

$$\arg\max_{\mathcal{M}} \mathcal{L} \approx \sum_{n=1:N} \log \mathbb{P}\left(O_n|\mathcal{M}, \mathcal{I}\right), \quad (2)$$

where NMS shoulders the responsibility to model the correlations in the entire sequence of detections. Since NMS is mostly[6] dependent on overlapping patterns, the information it can provide is limited compared to $\mathcal{O}_{0:n-1}$.

How can we do better? Inspired by networks that impose a memory [14, 18, 29, 35, 79] for sequential and reasoning tasks, and the two-dimensional nature of images, we propose to encode $\mathcal{O}_{0:n-1}$ in a spatial memory, where we learn to store all the previous detections. *I.e*., we introduce memory variable $\mathcal{S}_{n-1}$, which gets updated each time an object instance is detected, and the approximation becomes:

$$\arg\max_{\mathcal{M},\mathcal{S}} \mathcal{L} \approx \sum_{n=1:N} \log \mathbb{P}\left(O_n|\mathcal{S}_{n-1}, \mathcal{M}, \mathcal{I}\right), \quad (3)$$

where the memory $\mathcal{S}$ is jointly optimized with $\mathcal{M}$.

With the above formulation, the inference procedure for object becomes conditional: An empty memory is initialized at first (Sec. 4.1). Once an object instance is detected, selected cells (Sec. 4.2) in the memory gets updated (Sec. 4.4) with features (Sec. 4.3) extracted from the detected region. Then a context model (Sec. 4.5) aggregates spatial and other information from the memory, and outputs (Sec. 4.6) scores that help region proposal and region classification in FRCNN. Then the next potential detection is picked (Sec. 4.7) to update the memory again. This process goes on until a fixed number of iterations have reached (See Fig. 2 for an overview).

We now describe each module, beginning with a description of the memory itself.

4.1. Memory

Different from previous works that either mixes memory with computation [14, 18, 35] or mimics the one-dimensional memory in the Turing machine/von Neumann architecture [86], we would like to build a two-dimensional memory for images. This is intuitive because images are intrinsically 2D mappings of the 3D visual world. But more importantly, we aim to leverage the power of ConvNets for context reasoning, which "forces" us to provide an image-like 2D input.

How big the memory should be spatially? For object detection, FRCNN that operates entirely on `conv5_3` features can already retrieve even tiny objects (*e.g*. the ones in Fig. 1), suggesting that a resolution $1/16$ of the full image strikes a reasonable balance between speed and accuracy. At each location, the memory cell is a D=256 dimensional vector that stores the visual information discovered so far. Ideally, the initial values within the memory should capture the photographic bias of a natural image, *i.e*., prior about where a certain object tend to occur (*e.g. sun* is more likely to occur in the upper part). But the prior cannot be dependent on the input image size. To this end, we simply initialize the memory with a fixed spatial size (20×20×256 cells), and resize it according to the incoming `conv5_3` size using bilinear interpolation. In this way, the memory is fully utilized to learn the prior, regardless of different image sizes.

4.2. Indexing

The most difficult problem that previous works [29, 79] face when building an differentiable external memory is the design of memory indexing. The core problem is which memory cell to write to for what inputs. Luckily for spatial memory, there is a natural correspondence between memory and image. Specifically, the target regions to look up in 2D memory are already provided by proposals. Furthermore, RoI pooling [26, 40] is precisely the operations needed to *read* off from the spatial memory[7]. The only remaining task is to create a *write* function that updates the memory given a detection. This can be divided into two parts, "what" (Sec. 4.3), and "how" (Sec. 4.4).

4.3. Input Features

It may appear trivial, but the decision of what features to insert into the memory requires careful deliberation. First, since `conv5_3` feature preserves spatial information, we need to incorporate it. Specifically, we use `crop_and_resize` to obtain and resize the feature map to 14×14. This operation is similar to RoI [26] but with no max-pool. However, merely having `conv5_3` is not sufficient to capture the higher-level semantic information, especially pertaining which object class is detected. The detection score is particularly useful for disambiguation when two objects occur in the same region, *e.g*., a *person* riding a *horse*. Therefore, we also include `fc8` SoftMax score as an input, which is appended at each `conv5_3` locations and followed by two 1×1 `conv` layers to fuse the information (see Fig. 3). We choose the full score over a one-hot class vector, because it is more robust to false detections.

[6]Since NMS is applied in a per-class manner, there is also semantic information.

[7]Although RoI pooling only computes partial gradients, back-propagation w.r.t. bounding box coordinates are not entirely necessary [69] and previously found unstable [40].

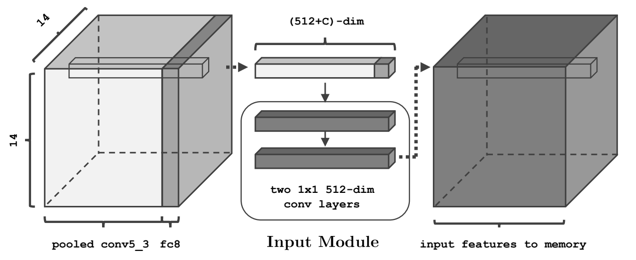

Figure 3: Illustration of the input module (Sec. 4.3). It assembles spatial and non-spatial features: detection scores after SoftMax (`fc8`) are tiled at each location of the RoI pooled 14×14 `conv5_3` feature. Two additional `conv` layers are used to merge the information from two sources. Dotted arrow shows how the feature at one location is transformed.

4.4. Writing

Given the region location and the input features $\mathbf{x}_n$, we update the corresponding memory cells with a convolutional Gated Recurrent Unit [14] (GRU), which uses 3×3 `conv` filters in place of `fc` layers as weights. The GRU has a reset gate, and an update gate, shared at each location and activated with Sigmoid function $\sigma(\cdot)$. Hyperbolic tangent $tanh(\cdot)$ is used to constrain the memory values between $-1.$ and 1. For alignment, the region from the original memory $\mathcal{S}_{n-1}$ is also cropped with the same RoI pooling operation to 14×14. After GRU, the new memory cells are placed back to $\mathcal{S}_n$ with the reverse operation of `crop_and_resize`.

4.5. Context Model

Now that the detected objects are encoded in the memory, all we have to do for context reasoning is stacking another ConNet on the top. In the current setup, we use a simple 5-layer all-convolutional network to extract the spatial patterns. Each `conv` filter has a spatial size of 3×3, and channel size of 256. Padding is added to keep the final layer `m-conv5` same size of `conv5_3`. To ease back-propagation, we add residual connections [33] every two layers.

4.6. Output

As for the module that outputs the reasoning results, we treat `m-conv5` exactly the same way as `conv5_3` in FR-CNN: 3 `conv` layers for region proposal, and 2 `fc` layers with RoI pooling for region classification. The `fc` layers have 2048 neurons each.

We design another residual architecture to combine the memory scores with the FRCNN scores (see Fig. 4): in the first iteration when the memory is empty, we only use FR-CNN for detection; from the second iteration on, we add the memory predictions on top of the FRCNN ones, so that the memory essentially provides the additional context to close the gap. This design allows a handy visualization of the prediction difference with/without context. But more importantly, such an architecture is critical to let us converge the full network. Details for this are covered in Sec. 5.1.

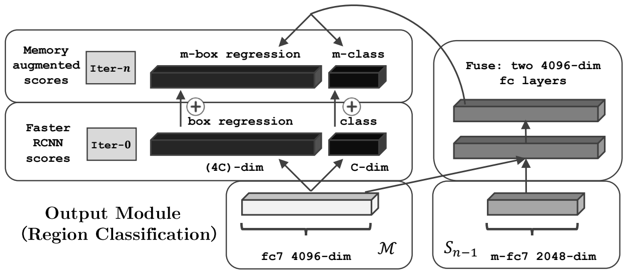

Figure 4: Illustration of the output module (Sec. 4.6) for region classification. FRCNN scores are optimized at the first iteration when memory is empty, and then augmented with memory scores in later iterations. Same is done for region proposals. Two additional `fc` layers are used to fuse FRCNN and memory features.

4.7. Selecting Next Region

Since spatial memory turns object detection into a sequential prediction problem, an important decision to make is which region to take-in next [7]. Intuitively, some objects are more useful serving as context for others (*e.g. person*) [21, 30, 32, 94], and some object instances are easier to detect and less prone to consequent errors. However, in this paper we simply follow a greedy strategy – the most confident foreground object box is selected to update the memory, leaving more advanced models that directly optimize the sequence [80] as future work.

5. Training the Spatial Memory

Like a standard network with recurrent connections, our SMN is trained by back-propagation through time (BPTT) [88], which unrolls the network multiple times before executing a weight-update. However, apart from the well-known gradient propagation issue, imposing the conditional structure on object detection incurs new challenges for training. Interestingly, the most difficult one we face in our experiment, is the "straightforward" task of de-duplication.

5.1. Learning De-duplication

Simply put, the functionality of de-duplication is: how can the network learn that a detected instance should no longer be detected again? More specifically, we need to design the output module (Sec. 4.6) to fuse the memory ($\mathcal{S}$) and FRCNN ($\mathcal{M}$) beliefs and predict intelligently: when the memory is empty, the FRCNN score should be used; but when the memory has the instance stored, the network needs to ignore, or *negate* the cue from FRCNN.

Since multi-layer networks are universal function approximators [38], our first attempt is to fuse the information by directly feeding into a multi-layer network (Fig. 5 (a)). However, joint-training fails to even converge FR-CNN. Suspicious that the longer, weaker supervision might

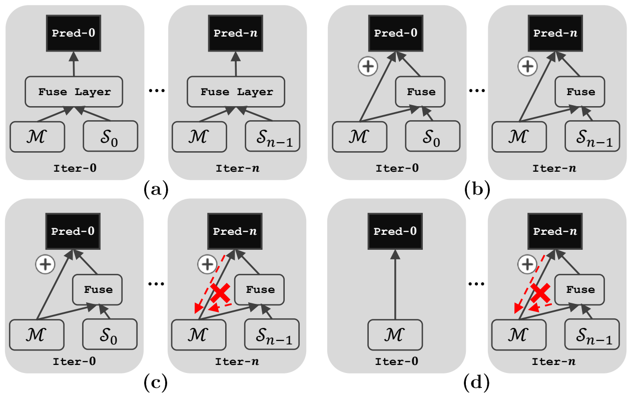

Figure 5: Four design choices for learning the functionality of de-duplication. $\mathcal{M}$ is FRCNN features, and $\mathcal{S}_{n-1}$ represents memory features. Each design is shown by two gray panels showing the information flow of Iteration 0 (left) and Iteration n>0 (right). We find it hard to even converge the network when the gradient is back-propagated to FRCNN in all iterations (a) & (b). Stop the gradient in later iterations (c) can successfully converge the network, and our final design (d) separates *perception* from *reasoning* and makes it easy to visualize the effect of context. All design choices are abstract and apply to both region proposal and classification. Please see Sec. 5.1 for more details.

be the cause, we also added skip connections [5] to guide the FRCNN training directly (Fig. 5 (b)). Yet it still does not help much. Tracking the learning process, we find where the actual problem lies – because the network needs to de-duplicate, it keeps receiving contradicting signals: the normal one that guides perception, and the adversarial one that prevents *more* perception. And because $\mathcal{S}$ also starts off from scratch, the signal it can provide is also weak and unreliable. As a result, part of both error signals are back-propagated to $\mathcal{M}$[8], causing trouble for learning further.

Realizing where the issue is, a direct solution is to just stop the adversarial signal from flowing back and canceling the normal one. Therefore, we stopped the gradient to FRCNN from second iteration on (Fig. 5 (c)), and the network can successfully converge.

To make it easy for training and showing the confidence changes for consequent detections given the context, we further reduced the architecture to exclude all memory related weights in the first iteration (Fig. 5 (d)). This way, the change in predictions with/without memory can be read-off directly[9], and training can be done separately for $\mathcal{M}$ and $\mathcal{S}$.

5.2. RoI Sampling

To avoid getting overwhelmed by negative boxes, FRCNN enforces a target sampling ratio for foreground/background boxes. The introduction of a spatial memory that learns to de-duplicate, brings in another special type – regions whose label is flipped from previous iterations. To keep these regions from being buried in negative examples too, we changed the sampling distribution to include flipped regions.

It is important to point out that RoI sampling greatly enhances the robustness of our sequential detection system. Because only $k \ll K$ regions are sampled from all regions, the overall most confident RoI is not guaranteed to be picked when updating the memory. This opens up chances for other highly confident boxes to be inserted into the sequence as well [80] and reduces over-fitting.

5.3. Multi-Tasking

We also practiced the idea of multi-task learning for SMN. The major motivation is to force the memory to memorize more: the basic SMN is only asked fulfill the mission of predicting the missing objects, which does not necessarily translate to a good memorization of previously detected objects. *E.g.*, it may remember that one region has an object in general, but does not store more categorical information beyond that. To better converge the memory, we also added a *reconstruction* loss [13, 70], *i.e.*, letting the network in addition predict the object classes it has stored in the memory. Specifically, we add an identical set of branches on top of the `m-conv5` features as FRCNN, for both region proposal and region classification in each iteration. These weights are used to predict only the previously detected objects.

5.4. Stage-wise Training

Thanks to the design of our memory augmented prediction, so far we have trained the full model in two separate stages, where FRCNN $\mathcal{M}$, the *perception* model can be optimized independently at first; then the *reasoning* model with spatial memory $\mathcal{S}$ is learned on top of fixed $\mathcal{M}$. This helps us isolate the influence of the base model and focus directly on the study of SMN.

For efficiency, we also follow a curriculum learning [8] strategy: bootstrap a SMN of more iterations (*e.g.* N=10) with a pre-trained SMN of fewer iterations (*e.g.* N=5). As N gets larger, the task becomes harder. Curriculum learning does not require re-learning de-duplication (which we learn with N from 2 to 4), and allows the network to focus more on object-object relationships instead.

5.5. Hyper-parameters

Given a pre-trained FRCNN or SMN (in the case of curriculum learning), we train a fixed number of 30k steps. The initial learning rate is set to $1e{-}3$ and reduced to $1e{-}4$ after 20k steps. Since we do not use automatic normalization tricks [41, 54], different variances are manually set when

[8]Since there are two sets of scores (from $\mathcal{M}$ and fused `fc`) added together for prediction in Fig. 5 (b), we find the conflicting signals are also propagated to the *biases* of these predictions: resulting in one going up and the other down while essentially canceling each other.

[9]Otherwise we have to run the inference again with $\mathcal{S}_0$.

Table 1: Baseline and initial analysis on COCO 2014 minival when constraining the number of detections N=5/10. AP and AR numbers are from COCO evaluation tool.

N	**Method**	AP	AR-10	AR-S	AR-M	AR-L
-	FRCNN [69]	24.2	33.7	11.7	39.5	54.1
-	Baseline [12]	29.1	38.7	17.7	44.9	56.9
N=5	Baseline	23.8	27.8	7.0	28.7	48.4
	SMN	**24.5**	**28.9**	**7.3**	**29.7**	**50.6**
N=10	Baseline	27.1	33.5	10.8	36.7	53.8
	SMN	**28.1**	**35.0**	**11.5**	**38.1**	**56.4**

initializing weights from scratch, in order to let different inputs contribute comparably (*e.g.* when concatenating `fc7` and `m-fc7`). Other hyper-parameters are kept the same to the ones used in FRCNN.

6. Experimental Results

We highlight the performance of our spatial memory network on COCO [52]. However, for ablative analysis and understanding the behaviour of our system, we use both PASCAL VOC 2007 [19] and COCO [52]. For VOC we use the *trainval* split for training, and *test* for evaluation. For COCO we use *trainval35k* [5] and *minival*. For evaluation, toolkits provided by the respective dataset are used. The main metrics (mAP, AP and AR) are based on detection average precision/recall.

Implementation Details: We use TensorFlow to implement our model, which is built on top of the open-sourced FRCNN implementation[10] serving as a baseline. For COCO, this implementation has an AP of 29.1% compared to the original one 24.2% [69].

Original FRCNN uses NMS for region sampling as well. However, NMS hurts our performance more since we do sequential prediction and one miss along the chain can negatively impact all the follow-up detections. To overcome this disadvantage, we would ideally like to examine all K regions in a sliding window fashion. However, due to the GPU memory limit, the top 5k regions are used instead. We analyze this choice in ablative analysis (Sec. 6.2). Due to the same limitation, our current implementation of SMN can only unroll N=10 times in a single GPU. At each timestep in SMN, we do a soft max-prediction for the top box selected, so that a single box can be assigned to multiple classes. We will also justify and analyze this choice in Sec. 6.2.

Initial Results: Table 1 shows the initial results of our approach as described. As it can be seen for N=5 detections per image our SMN give an AP of 24.5% and for N=10 if gives an AP of 28.1%. When the baseline is allowed the same number of detections (N=5, 10), the AP is 23.8% and 27.1%. Therefore, while we do outperform baseline for fixed number of detections per image, due to limited roll-out capability we are still ~1% below the baseline [12].

[10] https://github.com/endernewton/tf-faster-rcnn

Table 2: Final comparison between SMN and baselines. We additionally include MLP baseline where the number of parameters are kept the same as SMN for context aggregation and output. Top 5k regions are used to select proposal instead of NMS.

Method	AP	AP-.5	AP-.75	AP-S	AP-M	AP-L	AR-S	AR-M	AR-L
Baseline [12]	29.4	50.0	30.9	12.2	33.7	43.8	18.5	45.5	58.9
MLP	30.1	50.8	31.7	12.5	34.2	44.5	19.2	47.0	59.8
SMN	**31.6**	**52.2**	**33.2**	**14.4**	**35.7**	**45.8**	**20.5**	**48.8**	**63.2**

6.1. SMN for Hard Examples

In this section, we want to go beyond N=10 detections and see if the overall detection performance can be improved with SMN. Intuitively, for highly confident detections, ConvNet-based FRCNN is already doing a decent job and not much can be learned from an additional memory. It is the "tails" that need help from the context! This means two things: 1) with a limited resource budget, SMN should be used in later iterations to provide conditional information; and 2) at the beginning of the sequence, a standard FRCNN can work as a proxy. Given these insights, we experimented with the following strategy: For the first N_1 iterations, we use a standard FRCNN to detect easier objects and feed the memory with a sequence ordered by FRCNN confidence (after per-class NMS). Memory gets updated as objects come in, but does not output features to augment prediction. Only for the later N_2 iterations it acts normally as a context provider to detect harder examples. For COCO, we set N_1=50 and bootstrap from a N_2=10 SMN model.

Although SMN is trained with the goal of context reasoning and learns new functionality (*e.g.* de-duplication) that the original FRCNN does not have, it does have introduced more parameters for memory-augmented prediction. Therefore, we also add a MLP baseline, where a 5-layer ConvNet (Sec. 4.5) is directly stacked on top of `conv5_3` for context aggregation, and the same output modules (Sec. 4.6) are used to make predictions.

The results can be found in Table 2. As can be seen, on our final system, we are 2.2% better than the baseline FRCNN. This demonstrates our ability to find hard examples. It is worth noting that here *hard* does not necessarily translate to *small*. In fact, our reasoning system also helps big objects, potentially due to its ability to perform de-duplication more intelligently and benefit larger objects that are more likely to overlap.

Qualitative Results: We show a couple of examples of how context using spatial memory can help improve the performance and detections. In the first case, the score of *sheep* gets boosted due to other *sheep*. The score of *horse* decreases due to the detection of *cake* and *table*. Please check the supplementary material for more examples.

6.2. Ablative Analysis

We now perform ablative analysis to explain all our choices for the final implementation. For ablative analysis,

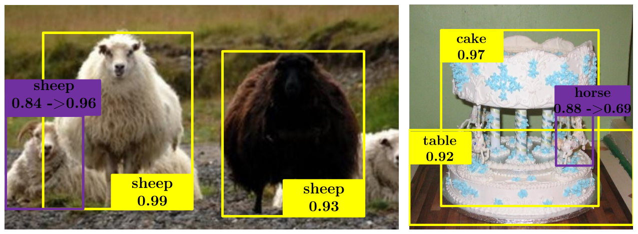

Figure 6: Examples of context has helped improve scores by reasoning. Left: the score of *sheep* is increased due to presence of other *sheep* in background. Right: the score of *horse* is decreased due to the detection of *cake* and *table*.

Table 3: Ablative analysis on VOC 2007 test and COCO 2014 minival. All approaches constrained by detections N=5/10. mAP is used to evaluate VOC, AP and AR numbers are from COCO.

N	**Method**	mAP	AP	AR-10	AR-S	AR-M	AR-L
N=5	Baseline (FR-CNN)	**65.8**	23.6	27.6	**7.0**	**29.1**	**47.4**
	SMN Base	63.6	23.3	27.2	6.7	28.0	46.1
	+ Sample Flipped	64.4	23.5	27.2	6.9	28.4	46.4
	SMN Full	64.6	**23.8**	**27.7**	6.9	28.5	**47.4**
N=10	Baseline	**70.3**	26.9	**33.2**	**10.9**	**36.6**	**52.7**
	SMN Full	67.5	26.6	32.6	10.3	35.6	52.1
	+Tune from N=5	67.8	**27.1**	32.7	10.3	35.9	52.3

we use both VOC and COCO datasets. The numbers are summarized in Table 3. For the comparisons shown here, we switch back to the standard NMS-based region sampling and select top k=300 RoIs as in original FRCNN. Also, when we do the roll-out, at each step we choose one detection and perform HardMax (rather than SoftMax): make the hard decision about what class does the selected box belong to – a natural idea for sequential prediction.

For N=5, we compared three models. First, SMN Base, where we simply train the network as is done in FRCNN. Next, regions with flipped labels (Sec. 5.2) are added to replaces some of the negative example – for training region proposal the ratio for positive/flipped/negative is 2:1:1, and for region classification it is 1:1:2. Third, SMN Full, where we keep the previous sampling strategy and in addition include the reconstruction loss (Sec. 5.3). Overall, both strategies help performance but with a seemly different strength: sampling flipped regions helps more on small objects, and multi-task learning helps more on bigger ones.

However, our best performance in Table 3 is still behind the baseline and judging from the COCO AR we believe the biggest issue lies in recall. Therefore, we take the best SMN Full model and conduct two other investigations specifically targeting recall. Here we only list the final results, please see supplementary material for more discussions.

SoftMax *vs.* HardMax: First, we address a subtle question: if we take top N detections with the memory and compare them directly with top N detections of Faster R-CNN: are these results comparable? It turns out to be not! As men-

Table 4: Investigating the recall issue. **S** stands for SoftMax based testing, and **H** for HardMax. $\not\mathbf{N}$ is short for Non-aggressive NMS, where top 5k RoIs are directly selected without NMS.

N	**Method**	$\not\mathbf{N}$	**Max**	mAP	AP	AR-10	AR-S	AR-M	AR-L
N=5	Baseline	✗	**S**	65.8	23.6	27.6	7.0	29.1	47.4
	SMN Full	✗	**S**	66.4	24.1	28.8	**7.5**	**29.7**	50.0
	Baseline	✗	**H**	65.4	23.5	27.2	6.7	28.6	46.9
	SMN Full	✗	**H**	64.6	23.8	27.7	6.9	28.5	47.4
	Baseline	✓	**S**	66.0	23.8	27.8	7.0	28.7	48.4
	SMN Full	✓	**S**	**66.6**	**24.5**	**28.9**	7.3	**29.7**	**50.6**
N=10	Baseline	✗	**S**	70.3	26.9	33.2	10.9	36.6	52.7
	SMN Full	✗	**S**	69.4	27.7	**35.0**	**11.6**	37.6	55.7
	Baseline	✗	**H**	68.0	26.4	31.9	9.7	35.0	50.7
	SMN Full	✗	**H**	67.8	27.1	32.7	10.3	35.9	52.3
	Baseline	✓	**S**	**70.4**	27.1	33.5	10.8	36.7	53.8
	SMN Full	✓	**S**	70.0	**28.1**	**35.0**	11.5	**38.1**	**56.4**

tioned in Sec. 3.4, because NMS is applied in a per-class manner, the actual number of box candidates it can put in the final detection is $k \times C$. To make it more clear, for a confusing region where *e.g.* the belief for *laptop* is 40% and *keyboard* is 35%, NMS can keep both candidates in the top N detections, whereas for SMN it can only keep the maximum one[11]. Therefore, to be fair, we try: a) HardMax for baseline; and b) SoftMax for SMN.

Non-aggressive NMS: Finally, we also evaluate our choice of non-aggressive NMS during RoI sampling. Both baseline and SMN perform better with 5k proposals; however our boost on AP is more significant due to sequential prediction issues.

7. Conclusion and Discussion

This paper is our first step towards instance-level reasoning in object detection with ConvNets. We introduce a simple yet powerful framework of spatial memory network, to model the instance-level context efficiently and effectively. Our spatial memory essentially assembles object instances back into a pseudo "image" representation. This memory can simply be fed into another ConvNet to extract context information and perform object-object relationship reasoning. We show our SMN direction is promising as it provides 2.2% improvement over baseline Faster RCNN on the COCO dataset with VGG16. We believe our framework is generic and should promote research focusing on knowledge-based reasoning on images.

Acknowledgements

XC would like to thank Jiwei Li for helpful discussions, and everyone who took time to provide useful feedback. This work was supported by ONR MURI N000141612007 and Sloan Fellowship to AG.

[11]It's also a result of our current input feature design, where we only used `fc8` and `conv5_3` features to update the memory without a top-down notion [75] of which class is picked, so there's no more need for SMN to return.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *TPAMI*, 34(11):2189–2202, 2012. 3

[2] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Learning to compose neural networks for question answering. *arXiv:1601.01705*, 2016. 2

[3] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh. Vqa: Visual question answering. In *ICCV*, 2015. 2

[4] M. Bar. Visual objects in context. *Nature Reviews Neuroscience*, 5(8):617–629, 2004. 1

[5] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016. 1, 6, 7

[6] M. Bellver, X. Giró-i Nieto, F. Marqués, and J. Torres. Hierarchical object detection with deep reinforcement learning. *arXiv:1611.03718*, 2016. 2

[7] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, 2015. 5

[8] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009. 6

[9] J. C. Caicedo and S. Lazebnik. Active object localization with deep reinforcement learning. In *ICCV*, 2015. 2

[10] P. Carbonetto, N. De Freitas, and K. Barnard. A statistical model for general contextual object recognition. In *ECCV*, 2004. 1

[11] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv:1606.00915*, 2016. 1

[12] X. Chen and A. Gupta. An implementation of faster rcnn with study for region sampling. *arXiv:1702.02138*, 2017. 3, 7

[13] X. Chen and C. Lawrence Zitnick. Mind's eye: A recurrent visual representation for image caption generation. In *CVPR*, 2015. 2, 6

[14] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*, 2014. 4, 5

[15] C. Desai, D. Ramanan, and C. C. Fowlkes. Discriminative models for multi-class object layout. *IJCV*, 95(1):1–12, 2011. 1, 2

[16] 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. 1

[17] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *arXiv:1411.4389*, 2014. 2

[18] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990. 4

[19] 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. 7

[20] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 32(9):1627–1645, 2010. 2, 3

[21] D. F. Fouhey, V. Delaitre, A. Gupta, A. A. Efros, I. Laptev, and J. Sivic. People watching: Human actions as a cue for single view geometry. *IJCV*, 110(3):259–274, 2014. 5

[22] C. Galleguillos and S. Belongie. Context based object categorization: A critical survey. *CVIU*, 114(6):712–722, 2010. 1

[23] C. Galleguillos, A. Rabinovich, and S. Belongie. Object categorization using co-occurrence, location and appearance. In *CVPR*, 2008. 1

[24] H. Gao, J. Mao, J. Zhou, Z. Huang, L. Wang, and W. Xu. Are you talking to a machine? dataset and methods for multilingual image question answering. In *NIPS*, 2015. 2

[25] S. Gidaris and N. Komodakis. Attend refine repeat: Active box proposal generation via in-out localization. *arXiv:1606.04446*, 2016. 2

[26] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 2, 3, 4

[27] G. Gkioxari, R. Girshick, and J. Malik. Contextual action recognition with r* cnn. In *ICCV*, 2015. 1

[28] A. Gonzalez-Garcia, A. Vezhnevets, and V. Ferrari. An active search strategy for efficient object class detection. In *CVPR*, 2015. 2

[29] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016. 4

[30] A. Gupta, A. Kembhavi, and L. S. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *TPAMI*, 31(10):1775–1789, 2009. 5

[31] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. *arXiv:1702.03920*, 2017. 3

[32] S. Gupta, B. Hariharan, and J. Malik. Exploring person context and local scene context for object detection. *arXiv:1511.08177*, 2015. 1, 5

[33] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 5

[34] P. Henderson and V. Ferrari. End-to-end training of object class detectors for mean average precision. *arXiv:1607.03476*, 2016. 2

[35] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 3, 4

[36] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *IJCV*, 80(1):3–15, 2008. 1

[37] A. Hollingworth. Does consistent scene context facilitate object perception? *Journal of Experimental Psychology: General*, 127(4):398, 1998. 1

[38] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. 5

[39] J. Hosang, R. Benenson, and B. Schiele. A convnet for non-maximum suppression. In *German Conference on Pattern Recognition*, 2016. 3

[40] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *arXiv:1611.10012*, 2016. 1, 3, 4

[41] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015. 6

[42] J. Johnson, A. Karpathy, and L. Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *CVPR*, 2016. 2

[43] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011. 1
[44] 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 crowdsourced dense image annotations. *arXiv:1602.07332*, 2016. 2
[45] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *TPAMI*, 31(12):2129–2142, 2009. 2
[46] C. Li, D. Parikh, and T. Chen. Extracting adaptive contextual cues from unlabeled regions. In *ICCV*, 2011. 1
[47] J. Li, Y. Wei, X. Liang, J. Dong, T. Xu, J. Feng, and S. Yan. Attentive contexts for object detection. *IEEE Transactions on Multimedia*, 2016. 3
[48] L.-J. Li, H. Su, L. Fei-Fei, and E. P. Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010. 2
[49] Y. Li, K. He, J. Sun, et al. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016. 1, 3
[50] X. Liang, L. Lee, and E. P. Xing. Deep variation-structured reinforcement learning for visual relationship and attribute detection. *arXiv:1703.03054*, 2017. 2
[51] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. *arXiv:1612.03144*, 2016. 1
[52] 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. 1, 2, 7
[53] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 1, 3
[54] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv:1506.04579*, 2015. 6
[55] J. Lu, J. Yang, D. Batra, and D. Parikh. Hierarchical question-image co-attention for visual question answering. In *NIPS*, 2016. 2
[56] Y. Lu, T. Javidi, and S. Lazebnik. Adaptive object detection using adjacency and zoom prediction. In *CVPR*, 2016. 2
[57] M. Malinowski, M. Rohrbach, and M. Fritz. Ask your neurons: A neural-based approach to answering questions about images. In *ICCV*, 2015. 2
[58] T. Malisiewicz and A. Efros. Beyond categories: The visual memex model for reasoning about object relationships. In *NIPS*, 2009. 1
[59] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv:1412.6632*, 2014. 2
[60] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *CVPR*, 2009. 1
[61] S. Mathe, A. Pirinen, and C. Sminchisescu. Reinforcement learning for visual object detection. In *CVPR*, 2016. 2
[62] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014. 1
[63] K. Murphy, A. Torralba, W. Freeman, et al. Using the forest to see the trees: a graphical model relating features, objects and scenes. *NIPS*, 2003. 1
[64] A. Oliva and A. Torralba. The role of context in object recognition. *Trends in cognitive sciences*, 11(12):520–527, 2007. 1
[65] t. E. Palmer. The effects of contextual scenes on the identification of objects. *Memory & Cognition*, 3(5):519–526, 1975. 1
[66] E. Parisotto and R. Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. *arXiv:1702.08360*, 2017. 3
[67] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *ICCV*, 2007. 1
[68] M. Ren and R. S. Zemel. End-to-end instance segmentation and counting with recurrent attention. *arXiv:1605.09410*, 2016. 3
[69] 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, 2, 3, 4, 7
[70] A. Rohrbach, M. Rohrbach, R. Hu, T. Darrell, and B. Schiele. Grounding of textual phrases in images by reconstruction. In *ECCV*, 2016. 6
[71] 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. 3
[72] K. J. Shih, S. Singh, and D. Hoiem. Where to look: Focus regions for visual question answering. In *CVPR*, 2016. 2
[73] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, 2006. 1
[74] A. Shrivastava and A. Gupta. Contextual priming and feedback for faster r-cnn. In *ECCV*, 2016. 1
[75] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond Skip Connections: Top-Down Modulation for Object Detection. *arXiv:1612.06851*, 2016. 8
[76] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014. 1, 3
[77] S. Singh, D. Hoiem, and D. Forsyth. Learning a sequential search for landmarks. In *CVPR*, 2015. 2
[78] R. Stewart, M. Andriluka, and A. Y. Ng. End-to-end people detection in crowded scenes. In *CVPR*, 2016. 3
[79] S. Sukhbaatar, J. Weston, R. Fergus, et al. End-to-end memory networks. In *NIPS*, 2015. 4
[80] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, 1999. 5, 6
[81] A. Torralba. Contextual priming for object detection. *IJCV*, 53(2):169–191, 2003. 1
[82] 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. 1
[83] 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. 1
[84] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. Sequence to sequence-video to text. In *ICCV*, 2015. 2
[85] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. *arXiv:1411.4555*, 2014. 2
[86] J. Von Neumann. First draft of a report on the edvac. *IEEE Annals of the History of Computing*, 15(4):27–75, 1993. 4
[87] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid.

Deepflow: Large displacement optical flow with deep matching. In *ICCV*, 2013. 1

[88] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. *Back-propagation: Theory, architectures and applications*, pages 433–486, 1995. 5

[89] S. Xie, X. Huang, and Z. Tu. Top-down learning for structured labeling with convolutional pseudoprior. In *ECCV*, 2016. 2

[90] C. Xiong, S. Merity, and R. Socher. Dynamic memory networks for visual and textual question answering. *arXiv*, 1603, 2016. 2

[91] H. Xu and K. Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *ECCV*, 2016. 2, 3

[92] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv:1502.03044*, 2015. 2, 3

[93] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola. Stacked attention networks for image question answering. In *CVPR*, 2016. 2

[94] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, 2010. 1, 5

[95] D. Yoo, S. Park, J.-Y. Lee, A. S. Paek, and I. So Kweon. Attentionnet: Aggregating weak directions for accurate object detection. In *ICCV*, 2015. 2

[96] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015. 2

[97] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei. Visual7w: Grounded question answering in images. In *CVPR*, 2016. 2

[98] 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