Relation Networks for Object Detection

Han Hu[1*] Jiayuan Gu[2*†] Zheng Zhang[1*] Jifeng Dai[1] Yichen Wei[1]

[1]Microsoft Research Asia
[2]Department of Machine Intelligence, School of EECS, Peking University
{hanhu,v-jiaygu,zhez,jifdai,yichenw}@microsoft.com

Abstract

Although it is well believed for years that modeling relations between objects would help object recognition, there has not been evidence that the idea is working in the deep learning era. All state-of-the-art object detection systems still rely on recognizing object instances ***individually****, without exploiting their relations during learning.*

This work proposes an object relation module. It processes a set of objects ***simultaneously*** *through interaction between their appearance feature and geometry, thus allowing modeling of their relations. It is lightweight and in-place. It does not require additional supervision and is easy to embed in existing networks. It is shown effective on improving object recognition and duplicate removal steps in the modern object detection pipeline. It verifies the efficacy of modeling object relations in CNN based detection. It gives rise to the* ***first fully end-to-end object detector****.*

1. Introduction

Recent years have witnessed significant progress in object detection using deep convolutional neutral networks (CNNs) [27]. The state-of-the-art object detection methods [24, 18, 38, 9, 32, 10, 23] mostly follow the *region based* paradigm since it is established in the seminal work R-CNN [19]. Given a sparse set of region proposals, object classification and bounding box regression are performed on each proposal *individually*. A *heuristic and hand crafted* post-processing step, non-maximum suppression (NMS), is then applied to remove duplicate detections.

It has been well recognized in the vision community for years that contextual information, or *relation* between objects, helps object recognition [12, 17, 46, 47, 39, 36, 17, 16, 6]. Most such works are before the prevalence of deep learning. During the deep learning era, there is no significant progress about exploiting object relation for detection learning. Most methods still focus on recognizing objects separately.

One reason is that object-object relation is hard to model. The objects are at arbitrary image locations, of different scales, within different categories, and their number may vary across different images. The modern CNN based methods mostly have a simple regular network structure [25, 23]. It is unclear how to accommodate above irregularities in existing methods.

Our approach is motivated by the success of attention modules in natural language processing field [5, 49]. An attention module can effect an individual element (*e.g.*, a word in the target sentence in machine translation) by aggregating information (or features) from a set of elements (*e.g.*, all words in the source sentence). The aggregation weights are automatically learnt, driven by the task goal. An attention module can model dependency between the elements, without making excessive assumptions on their locations and feature distributions. Recently, attention modules have been successfully applied in vision problems such as image captioning [50].

In this work, for the first time we propose an adapted attention module for object detection. It is built upon a basic attention module. An apparent distinction is that the primitive elements are objects instead of words. The objects have 2D spatial arrangement and variations in scale/aspect ratio. Their locations, or geometric features in a general sense, play a more complex and important role than the word location in an 1D sentence. Accordingly, the proposed module extends the original attention weight into two components: the original weight and a new geometric weight. The latter models the spatial relationships between objects and only considers the *relative geometry* between them, making the module *translation invariant*, a desirable property for object recognition. The new geometric weight proves important in our experiments.

The module is called *object relation module*. It shares the same advantages of an attention module. It takes variable number of inputs, runs in parallel (as opposed to sequential relation modeling [29, 44, 6]), is fully differen-

*Equal contribution. †This work is done when Jiayuan Gu is an intern at Microsoft Research Asia.

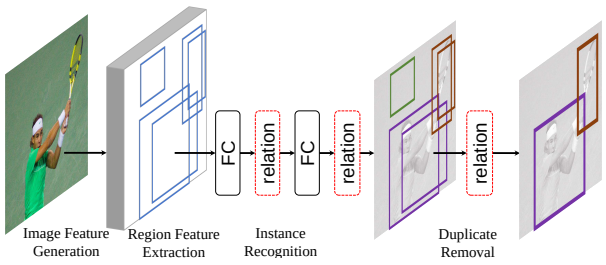


Figure 1. Current state-of-the-art object detectors are based on a four-step pipeline. Our object relation module (illustrated as red dashed boxes) can be conveniently adopted to improve both instance recognition and duplicate removal steps, *resulting in an end-to-end object detector*.

tiable and is in-place (no dimension change between input and output). As a result, it serves as a basic building block that is usable in any architecture flexibly.

Specifically, it is applied to several state-of-the-art object detection architectures [38, 10, 32] and show consistent improvement. As illustrated in Figure 1, it is applied to improve the *instance recognition* step and learn the *duplicate removal* step (see Section 4.1 for details). For instance recognition, the relation module enables joint reasoning of all objects and improves recognition accuracy (Section 4.2). For duplicate removal, the traditional NMS method is replaced and improved by a lightweight relation network (Section 4.3), resulting in *the first end-to-end object detector* (Section 4.4), to our best knowledge.

In principle, our approach is fundamentally different from and would complement most (if not all) CNN based object detection methods. It exploits a new dimension: *a set of objects are processed, reasoned and affect each other simultaneously, instead of recognized individually*.

The object relation module is general and not limited to object detection. We do not see any reason preventing it from finding broader applications in vision tasks, such as instance segmentation [30], action recognition [41], object relationship detection [28], caption [50], VQA [1], etc.

2. Related Works

Object relation in post-processing Most early works use object relations as a post-processing step [12, 17, 46, 47, 36, 17]. The detected objects are re-scored by considering object relationships. For example, *co-occurrence*, which indicates how likely two object classes can exist in a same image, is used by DPM [15] to refine object scores. The subsequent approaches [7, 36] try more complex relation models, by taking additional *position* and *size* [3] into account. We refer readers to [16] for a more detailed survey. These methods achieve moderate success in pre-deep learning era but do not prove effective in deep ConvNets. A possible reason is that deep ConvNets have implicitly incorporated contextual information by the large receptive field.

Sequential relation modeling Several recent works perform sequential reasoning (LSTM [29, 44] and spatial memory network (SMN) [6]) to model object relations. During detection, objects detected earlier are used to help finding objects next. Training in such methods is usually sophisticated. More importantly, they do not show evidence of improving the state-of-the-art object detection approaches, which are simple feed-forward networks.

In contrast, our approach is parallel for multiple objects. It naturally fits into and improves modern object detectors.

Human centered scenarios Quite a few works focus on *human-object* relation [51, 22, 20, 21]. They usually require additional annotations of relation, such as human action. In contrast, our approach is general for object-object relation and does not need additional supervision.

Duplicate removal In spite of the significant progress of object detection using deep learning, the most effective method for this task is still the greedy and hand-crafted non-maximum suppression (NMS) and its soft version [4]. This task naturally needs relation modeling. For example, NMS uses simple relations between bounding boxes and scores.

Recently, GossipNet [26] attempts to learn *duplicate removal* by processing a set of objects as a whole, therefore sharing the similar spirit of ours. However, its network is specifically designed for the task and very complex (depth>80). Its accuracy is comparable to NMS but computation cost is demanding. Although it allows end-to-end learning in principle, no experimental evidence is shown.

In contrast, our relation module is simple, general and applied to duplicate removal as an application. Our network for duplicate removal is much simpler, has small computation overhead and surpasses SoftNMS [4]. More importantly, we show that an end-to-end object detection learning is feasible and effective, *for the first time*.

Algorithm 1 Object relation module. Input is N objects $\{(f_A^n, f_G^n)\}_{n=1}^N$. Dimension of appearance feature f_A is d_f. After each algorithm line is the computation complexity.

hyper param: number of relations N_r
hyper param d_k: key feature dimension
hyper param d_g: geometric feature embedding dimension
learnt weights: $\{W_K^r, W_Q^r, W_V^r, W_G^r\}_{r=1}^{N_r}$
for every (n, r) **do** $\triangleright\ O(NN_r)$
  compute $\{\omega_G^{mn,r}\}_{m=1}^N$ using Eq. (5) $\triangleright\ O(Nd_g)$
  compute $\{\omega_A^{mn,r}\}_{m=1}^N$ using Eq. (4) $\triangleright\ O(d_k(2d_f + N))$
  compute $\{\omega^{mn,r}\}_{m=1}^N$ using Eq. (3) $\triangleright\ O(N)$
  compute $f_R^r(n)$ using Eq. (2) $\triangleright\ O(d_f^2/N_r + Nd_f/N_r)$
end for
output new feature $\{f_A^n\}_{n=1}^N$ using Eq. (6)

3. Object Relation Module

We first review a basic attention module, called "Scaled Dot-Product Attention" [49]. The input consists of queries and keys of dimension d_k, and values of dimension d_v. Dot

product is performed between the query and all keys to obtain their similarity. A softmax function is applied to obtain the weights on the values. Given a query $\mathbf{q}$, all keys (packed into matrices K) and values (packed into V), the output value is weighted average over input values,

$$v^{out} = softmax(\frac{\mathbf{q}K^t}{\sqrt{d_k}})V. \quad (1)$$

We now describe object relation computation. Let an object consists of its *geometric* feature $\mathbf{f}_G$ and appearance feature $\mathbf{f}_A$. In this work, $\mathbf{f}_G$ is simply a 4-dimensional object bounding box and $\mathbf{f}_A$ is up to the task (Section 4.2 and 4.3).

Given input set of N objects $\{(\mathbf{f}_A^n, \mathbf{f}_G^n)\}_{n=1}^N$, the *relation feature* $\mathbf{f}_R(n)$ of the whole object set with respect to the n^{th} object, is computed as

$$\mathbf{f}_R(n) = \sum_m \omega^{mn} \cdot (W_V \cdot \mathbf{f}_A^m). \quad (2)$$

The output is a weighted sum of appearance features from other objects, linearly transformed by W_V (corresponding to values V in Eq. (1)). The relation weight ω^{mn} indicates the impact from other objects. It is computed as

$$\omega^{mn} = \frac{\omega_G^{mn} \cdot \exp(\omega_A^{mn})}{\sum_k \omega_G^{kn} \cdot \exp(\omega_A^{kn})}. \quad (3)$$

Appearance weight ω_A^{mn} is computed as dot product, similarly as in Eq. (1),

$$\omega_A^{mn} = \frac{dot(W_K \mathbf{f}_A^m, W_Q \mathbf{f}_A^n)}{\sqrt{d_k}}. \quad (4)$$

Both W_K and W_Q are matrices and play a similar role as K and Q in Eq. (1). They project the original features $\mathbf{f}_A^m$ and $\mathbf{f}_A^n$ into subspaces to measure how well they match. The feature dimension after projection is d_k.

Geometry weight is computed as

$$\omega_G^{mn} = \max\{0, W_G \cdot \mathcal{E}_G(\mathbf{f}_G^m, \mathbf{f}_G^n)\}. \quad (5)$$

There are two steps. First, the geometry features of the two objects are embedded to a high-dimensional representation, denoted as $\mathcal{E}_G$. To make it invariant to translation and scale transformations, a 4-dimensional relative geometry feature is used, as $\left(\log(\frac{|x_m - x_n|}{w_m}), \log(\frac{|y_m - y_n|}{h_m}), \log(\frac{w_n}{w_m}), \log(\frac{h_n}{h_m})\right)^T$[1]. This 4-d feature is embedded to a high-dimensional representation by method in [49], which computes cosine and sine functions of different wavelengths. The feature dimension after embedding is d_g.

[1]It is a modified version of the widely used bounding box regression target [19]. The first two elements are transformed using $\log(\cdot)$ to count more on close-by objects. The intuition behind this modification is that we need to model distant objects while original bounding box regression only considers close-by objects.

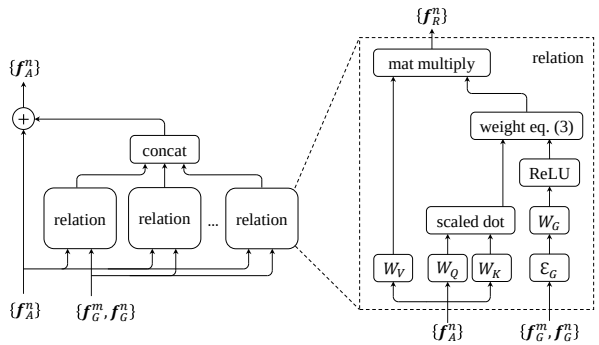


Figure 2. **Left**: object relation module as Eq. (6); **Right**: relation feature computation as Eq. (2).

Second, the embedded feature is transformed by W_G into a scalar weight and trimmed at 0, acting as a ReLU nonlinearity. The zero trimming operation restricts relations only between objects of certain geometric relationships.

The usage of geometric weight Eq. (5) in the attention weight Eq. (3) makes our approach distinct from the basic attention Eq. (1). To validate the effectiveness of Eq. (5), we also experimented with two other simpler variants. The first is called *none*. It does not use geometric weight Eq. (5). ω_G^{mn} is a constant 1.0 in Eq. (3). The second is called *unary*. It follows the recent approaches [13, 49]. Specifically, $\mathbf{f}_G$ is embedded into a high-dimension (same as $\mathbf{f}_A$) space in the same way [49] and added onto $\mathbf{f}_A$ to form the new appearance feature. The attention weight is then computed as *none* method. The effectiveness of our geometry weight is validated in Table 1(a) and Section 5.2.

An *object relation module* aggregates in total N_r relation features and augments the input object's appearance feature via addition,

$$\mathbf{f}_A^n = \mathbf{f}_A^n + \text{Concat}[\mathbf{f}_R^1(n), ..., \mathbf{f}_R^{N_r}(n)], \text{for all } n. \quad (6)$$

Concat$(\cdot)$ is used to aggregate multiple relation features[2]. To match the channel dimension, the output channel of each W_V^r is set as $\frac{1}{N_r}$ of the dimension of input feature $\mathbf{f}_A^m$.

The object relation module Eq. (6) is summarized in Algorithm 1. It is easy to implement using basic operators, as illustrated in Figure 2.

Each relation function in Eq. (2) is parameterized by four matrices (W_K, W_Q, W_G, W_V), in total $4N_r$. Let d_f be the dimension of input feature $\mathbf{f}_A$. The number of parameters is

$$O(\text{Space}) = N_r(2d_f d_k + d_g) + d_f^2. \quad (7)$$

Following Algorithm 1, the computation complexity is

$$O(\text{Comp.}) = Nd_f(2N_r d_k + d_f) + N^2 N_r(d_g + d_k + d_f/N_r + 1). \quad (8)$$

[2]An alternative is Addition$(\cdot)$. However, its computation cost would be much higher because we have to match the channel dimensions of two terms in Eq. (6). Only Concat$(\cdot)$ is experimented in this work.

Typical parameter value is $N_r = 16$, $d_k = 64$, $d_g = 64$. In general, N and d_f are usually at the scale of hundreds. The overall computation overhead is low when applied to modern object detectors.

The relation module has the same input and output dimension, and hence can be regarded as a basic building block to be used in-place within any network architecture. It is fully differentiable, and thus can be easily optimized with back-propagation. Below it is applied in modern object detection systems.

4. Relation Networks For Object Detection

4.1. Review of Object Detection Pipeline

This work conforms to the *region based* object detection paradigm. The paradigm is established in the seminal work R-CNN [19] and includes majority of modern object detectors [24, 18, 38, 9, 32, 10, 23][3]. A four step pipeline is used in all previous works, as summarized here.

First step generates full image features. From the input image, a deep convolutional *backbone* network extracts full resolution convolutional features (usually $16\times$ smaller than input image resolution). The backbone network [42, 45, 43, 25, 8, 53] is pre-trained on ImageNet classification task [11] and fine-tuned during detection training.

Second step generates regional features. From the convolutional features and a sparse set of region proposals [48, 52, 38], a RoI pooling layer [24, 18, 23] extracts fixed resolution regional features (*e.g.*, 7×7) for each proposal.

Third step performs instance recognition. From each proposal's regional features, a *head* network predicts the probabilities of the proposal belonging to certain object categories, and refine the proposal bounding box via regression. This network is usually shallow, randomly initialized, and jointly trained together with backbone network during detection training.

Last step performs duplicate removal. As each object should be detected only once, duplicated detections on the same object should be removed. This is usually implemented as a heuristic post-processing step called *non-maximum suppression* (NMS). Although NMS works well in practice, it is manually designed and sub-optimal. It prohibits the end-to-end learning for object detection.

In this work, the proposed object relation module is used in the last two steps. We show that it enhances the *instance recognition* (Section 4.2) and learns *duplicate removal* (Section 4.3). Both steps can be easily trained, either independently or jointly (Section 4.4). The joint training further boosts the accuracy and gives rise to *the first end-to-end general object detection system*.

Our implementation of different architectures To validate the effectiveness and generality of our approach, we experimented with different combination of state-of-the-art *backbone* networks (ResNet [25]), and best-performing detection architectures including faster RCNN [38], feature pyramid networks (FPN) [32], and deformable convolutional network (DCN) [10]. Region proposal network (RPN) [38] is used to generate proposals.

- *Faster RCNN* [38]. It is directly built on backbone networks such as ResNet [25]. Following [38], RPN is applied on the conv4 feature maps. Following [10], the instance recognition head network is applied on a new 256-d 1×1 convolution layer added after conv5, for dimension reduction. Note that the stride in conv5 is changed from 2 to 1, as common practice [25].

- *FPN* [32]. Compared to Faster RCNN, it modifies the backbone network by adding top-down and lateral connections to build a feature pyramid that facilitates end-to-end learning across different scales. RPN and head networks are applied on features of all scales in the pyramid. We follow the training details in [32].

- *DCN* [10]. Compared to Faster RCNN, it modifies the backbone network by replacing the last few convolution layers in conv5 by deformable convolution layers. It also replace the standard RoI pooling by deformable RoI pooling. We follow the training details in [10].

Despite the differences, a commonality in above architectures is that they all adopt the same head network structure, that is, the RoI pooled regional features undergo two fully connected layers (*2fc*) to generate the final features for proposal classification and bounding box regression.

Below, we show that relation modules can enhance the instance recognition step using the *2fc* head.

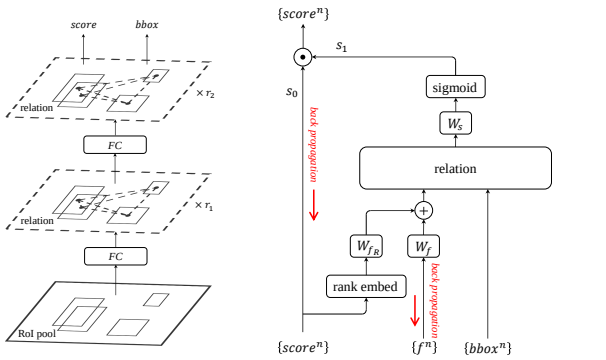
4.2. Relation for Instance Recognition

Given the RoI pooled features for n^{th} proposal, two fc layers with dimension 1024 are applied. The instance classification and bounding box regression are then performed via linear layers. This process is summarized as

$$\begin{aligned} RoI\_Feat_n &\xrightarrow{FC} 1024 \\ &\xrightarrow{FC} 1024 \\ &\xrightarrow{LINEAR} (score_n, bbox_n) \end{aligned} \tag{9}$$

The object relation module (Section 3, Algorithm 1) can transform the 1024-d features of all proposals without changing the feature dimension. Therefore, it can be used

[3] Another object detection paradigm is based on *dense sliding windows* [35, 37, 33]. In this paradigm, the object number N is much larger. Directly applying relation module as in this work is computationally costly. How to effectively model relations between dense objects is yet unclear.



(a) enhanced *2fc* head (b) duplicate removal network

Figure 3. Illustration of enhanced *2fc* head (a) and duplicate classification network (b) by object relation modules.

after either fc layer in Eq. (9) for arbitrary number of times[4]. Such enhanced *2fc+RM* (RM for relation module) head is illustrated in Figure 3 (a) and summarized as

$$\begin{aligned} \{RoI\_Feat_n\}_{n=1}^{N} &\xrightarrow{FC} 1024 \cdot N \xrightarrow{\{RM\}^{r_1}} 1024 \cdot N \\ &\xrightarrow{FC} 1024 \cdot N \xrightarrow{\{RM\}^{r_2}} 1024 \cdot N \\ &\xrightarrow{LINEAR} \{(score_n, bbox_n)\}_{n=1}^{N} \end{aligned} \tag{10}$$

In Eq. (10), r_1 and r_2 indicate how many times a relation module is repeated. Note that a relation module also needs all proposals' bounding boxes as input. This notation is neglected here for clarify.

Adding relation modules can effectively enhance the instance recognition accuracy. This is verified via comprehensive ablation studies in experiments (Section 5.1).

4.3. Relation for Duplicate Removal

The task of duplicate removal naturally requires exploiting the relation between objects. The heuristic NMS method is a simple example: the object with the highest score will erase its nearby objects (geometric relation) with inferior scores (score relation).

In spite of its simplicity, the greedy nature and manually chosen parameters in NMS makes it a clear sub-optimal choice. Below we show that the proposed relation module can learn to remove duplicate in a manner that is simple as well but more effective.

Duplicate removal is a two class classification problem. For each ground truth object, only one detected object matched to it is classified as *correct*. Others matched to it are classified as *duplicate*.

This classification is performed via a network, as illustrated in Figure 3 (b). The input is a set of detected objects (output from instance recognition, Eq. (9) or (10)). Each object has its final 1024-d feature, the classification score s_0, and bounding box. The network outputs a binary classification probability $s_1 \in [0, 1]$ (1 for *correct* and 0 for *duplicate*) for each object. The multiplication of two scores $s_0 s_1$ is the final classification score. Therefore, a good detection should have both scores large.

The network has three steps. First, the 1024-d feature and classification score is fused to generate the appearance feature. Second, a relation module transforms such appearance features of all objects. Last, the transformed features of each object pass a linear classifier (W_s in Figure 3 (b)) and sigmoid to output the probability $\in [0, 1]$.

The relation module is at the core of the network. It enables effective end-to-end learning using information from multiple sources (the bounding boxes, original appearance features and classification scores). In addition, the usage of the classification scores also turns out important.

Rank feature We found that it is most effective to transform the score into a rank, instead of using its value. Specifically, the input N objects are sorted in descending order of their scores. Each object is given a rank $\in [1, N]$ accordingly. The scalar rank is then embedded into a higher dimensional 128-d feature, using the same method [49] as for geometry feature embedding in Section 3.

Both the rank feature and original 1024-d appearance feature are transformed to 128-d (via W_{f_R} and W_f in Figure 3 (b), respectively), and added as the input to the relation module.

Which object is correct? Given a number of detected objects, it is not immediately clear which one should be matched to a ground truth object as *correct*. The most obvious choice would be following the evaluation criterion of Pascal VOC [14] or COCO datasets [34]. That is, given a predefined threshold η for the IoU between detection box and ground truth box, all detection boxes with IoU $\geq \eta$ are firstly matched to the same ground truth. The detection box with highest score is *correct* and others are *duplicate*.

Consequently, such selection criteria work best when learning and evaluation use the same threshold η . For example, using $\eta = 0.5$ in learning produces best mAP@0.5 metric but not mAP @0.75. This is verified in Table 4.

This observation suggests a unique benefit of our approach that is missing in NMS: the duplicate removal step can be adaptively learnt according to needs, instead of using preset parameters. For example, a large η should be used when a high localization accuracy is desired.

Motivated by the COCO evaluation criteria (mAP@$0.5-0.95$), our best practice is to use multiple thresholds simultaneously, i.e., $\eta \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. Specifically, the classifier W_s in Figure. 3 (b) is changed to output multiple probabilities corresponding to different IoU thresholds and *correct* detections, resulting in multiple binary classi-

[4]The relation module can also be used directly on the regional features. The high dimension ($256 \times 7^2 = 12544$ in our implementation), however, introduces large computational overhead. We did not do this experiment.

fication loss terms. The training is well balanced between different cases. During inference, the multiple probabilities are simply averaged as a single output.

Training The binary cross entropy loss is used on the final score (multiplication of two scores, see Figure 3 (b)). The loss is averaged over all detection boxes on all object categories. A single network is trained for all object categories.

Note that the duplicate classification problem is extremely imbalanced. Most detections are *duplicate*. The ratio of *correct* detections is usually < 0.01. Nevertheless, we found the simple cross entropy loss works well. This is attributed to the multiplicative behavior in the final score $s_0 s_1$. Because most detections have very small s_0 (mostly < 0.01) and thus small $s_0 s_1$. The magnitude of their loss values $L = -\log(1 - s_0 s_1)$ (for non-*correct* object) and back-propagated gradients $\partial L / \partial s_1 = s_0/(1 - s_0 s_1)$ is also very small and does not affect the optimization much. Intuitively, training is focused on a few real *duplicate* detections with large s_0. This shares the similar spirit to the recent focal loss work [31], where majority insignificant loss terms are down weighted and play minor roles during optimization.

Inference The same duplicate removal network is applied for all object categories independently. At a first glance, the runtime complexity could be high, when the number of object classes (80 for COCO dataset [34]) and detections ($N = 300$) is high. Nevertheless, in practice most detections' original score s_0 is nearly 0 in most object classes. For example, in the experiments in Table 4, only 12.0% classes have detection scores > 0.01 and in these classes only 6.8% detections have scores > 0.01.

After removing these insignificant classes and detections, the final recognition accuracy is not affected. Running the duplicate removal network on remaining detections is practical, taking about 2 ms on a Titan X GPU. Note that NMS and SoftNMS [4] methods are sequential and take about 5 ms on a CPU [4]. Also note that the recent learning NMS work [26] uses a very deep and complex network (depth up to 80), which is much less efficient than ours.

4.4. End-to-End Object Detection

The duplicate removal network is trained alone in Section 4.3. Nevertheless, there is nothing preventing the training to be end-to-end. As indicated by the red arrows in Figure 3 (b), the back propagated gradients can pass into the original 1024-d features and classification scores, which can further propagate back into the *head* and *backbone* networks.

Our end-to-end training simply combines the region proposal loss, the instance recognition loss in Section 4.2 and duplicate classification loss in Section 4.3, with equal weights. For instance recognition, either the original head Eq. (9) or enhanced head Eq. (10) can be used.

The end-to-end training is clearly feasible, but does it work? At a first glance, there are two issues.

First, the goals of instance recognition step and duplicate removal step seem contradictory. The former expects all objects matched to the same ground truth object to have high scores. The latter expects only one of them does. In our experiment, we found the end-to-end training works well and converges equally fast for both networks, compared to when they are trained individually as in Section 4.2 and 4.3. We believe this seemingly conflict is reconciled, again, via the multiplicative behavior in the final score $s_0 s_1$, which makes the two goals complementary other than conflicting. The instance recognition step only needs to produce high score s_0 for good detections (no matter duplicate or not). The duplicate removal step only needs to produce low score s_1 for duplicates. The majority non-object or *duplicate* detection is correct as long as one of the two scores is correct.

Second, the binary classification ground truth label in the duplicate removal step depends on the output from the instance recognition step, and changes during the course of end-to-end training. However, in experiments we did not observe adverse effects caused by this instability. While there is no theoretical evidence yet, our guess is that the duplicate removal network is relatively easy to train and the instable label may serve as a means of regularization.

As verified in experiments (Section 5.3), the end-to-end training improves the recognition accuracy.

5. Experiments

All experiments are performed on COCO detection datasets with 80 object categories [34]. A union of $80k$ train images and a $35k$ subset of val images are used for training [2, 32]. Most ablation experiments report detection accuracies on a subset of $5k$ unused val images (denoted as *minival*) as common practice [2, 32]. Table 5 also reports accuracies on $test\text{-}dev$ for system-level comparison.

For backbone networks, we use ResNet-50 and ResNet-101 [25]. Unless otherwise noted, ResNet-50 is used.

For Faster RCNN [38] and DCN [10], our training mostly follow [10]. For FPN [32], our training mostly follow [32].

5.1. Relation for Instance Recognition

In this section, NMS with IoU threshold of 0.6 is used for duplicate removal for all experiments.

Relation module improves instance recognition Table 1 compares the baseline $2fc$ head in Eq. (9) with the proposed $2fc + RM$ head in Eq. (10), under various parameters.

We firstly note that our baseline implementation achieves reasonable accuracy (29.6 mAP) when compared with the literature (*e.g.*, [32] reports 28.0 using ResNet-50 and [10] reports 29.4 using ResNet-101).

2fc baseline	***(a)**: usage of geometric feature*			***(b)**: number of relations N_r*						***(c)**: number of relation modules $\{r_1, r_2\}$*				
	none	unary	ours*	1	2	4	8	16*	32	$\{1,0\}$	$\{0,1\}$	$\{1,1\}$*	$\{2,2\}$	$\{4,4\}$
29.6	30.3	31.1	**31.9**	30.5	30.6	31.3	31.7	**31.9**	31.7	31.7	31.4	31.9	32.5	**32.8**

Table 1. Ablation study of relation module structure and parameters (* for default). mAP@all is reported.

head	mAP	mAP_{50}	mAP_{75}	# params	# FLOPS
(a) 2fc (1024)	29.6	50.9	30.1	38.0M	80.2B
(b) 2fc (1432)	29.7	50.3	30.2	44.1M	82.0B
(c) 3fc (1024)	29.0	49.4	29.6	39.0M	80.5B
(d) 2fc+res $\{r_1, r_2\}$=$\{1, 1\}$	29.9	50.6	30.5	44.0M	82.1B
(e) 2fc (1024) + global	29.6	50.3	30.8	38.2M	82.2B
(f) 2fc+RM $\{r_1, r_2\}$=$\{1, 1\}$	**31.9**	53.7	33.1	44.0M	82.6B
(g) 2fc (1024) + 2×	30.4	51.7	31.4	50.2M	83.8B
(h) 2fc+2×+RM $\{r_1, r_2\}$=$\{1, 1\}$	**32.5**	54.3	34.1	56.2M	86.2B
(i) 2fc+res $\{r_1, r_2\}$=$\{2, 2\}$	29.8	50.5	30.5	50.0M	84.0B
(j) 2fc+RM $\{r_1, r_2\}$=$\{2, 2\}$	**32.5**	54.0	33.8	50.0M	84.9B

Table 2. Comparison of various heads with similar complexity.

NMS	ours	rank f_R		appearance f	geometric *bbox*	
	$\{f_R, f, bbox\}$	*none*	s_0	*none*	*none*	*unary*
29.6	**30.3**	26.6	28.3	29.9	28.1	28.2

Table 3. Ablation study of input features for duplicate removal network (*none* indicates without such feature).

method	parameters	mAP	mAP_{50}	mAP_{75}
NMS	$N_t = 0.3$	29.0	51.4	29.4
NMS	$N_t = 0.4$	29.4	**52.1**	29.5
NMS	$N_t = 0.5$	29.6	51.9	29.7
NMS	$N_t = 0.6$	**29.6**	50.9	30.1
NMS	$N_t = 0.7$	28.4	46.6	**30.7**
SoftNMS	$\sigma = 0.2$	30.0	**52.3**	30.5
SoftNMS	$\sigma = 0.4$	30.2	51.7	31.3
SoftNMS	$\sigma = 0.6$	**30.2**	50.9	31.6
SoftNMS	$\sigma = 0.8$	29.9	49.9	**31.6**
SoftNMS	$\sigma = 1.0$	29.7	49.7	31.6
ours	$\eta = 0.5$	30.3	**51.9**	31.5
ours	$\eta = 0.75$	30.1	49.0	**32.7**
ours	$\eta \in [0.5, 0.9]$	**30.5**	50.2	32.4
ours (e2e)	$\eta \in [0.5, 0.9]$	**31.0**	51.4	32.8

Table 4. Comparison of NMS methods and our approach (Section 4.3). Last row uses end-to-end training (Section 4.4).

Ablation studies are performed on three key parameters.

Usage of geometric feature. As analyzed in Section 3, our usage of geometric feature in Eq. (5) is compared to two plain implementations. Results show that our approach is the best, although all the three surpass the baseline.

Number of relations N_r. Using more relations steadily improves the accuracy. The improvement saturates at $N_r = 16$, where +2.3 mAP gain is achieved.

Number of modules. Using more relation modules steadily improves accuracy, up to +3.2 mAP gain. As this also increases the parameter and computation complexity, by default $r_1 = 1, r_2 = 1$ is used.

Does the improvement come from more parameters or depths? Table 2 answers this question by enhancing the baseline $2fc$ head (a) in width or depth such that its complexity is comparable to that of adding relation modules.

A wider 2fc head (1432-d, b) only introduces small improvement (+0.1 mAP). A deeper 3fc head (c) deteriorates the accuracy (-0.6 mAP), probably due to the difficulty of training. To make the training easier, residual blocks [25] are used[5] (d), but only moderate improvement is observed (+0.3 mAP). When global context is used (e, 2048-d global average pooled features are concatenated with the second 1024-d instance feature before classification), no improvement is observed. By contrast, our approach (f) significantly improves the accuracy (+2.3 mAP).

We also consider another baseline which concatenates the original pooled features with the ones from a $2\times$ larger RoI (g), the performance is improved from 29.6 to 30.4 mAP, indicating a better way of utilizing context cues. In addition, we combine this new head with relation modules, that is, replacing the *2fc* with $\{r_1, r_2\}$=$\{1, 1\}$ (h). We get 32.5 mAP, which is 0.6 better than setting (f) (31.9 mAP). This indicates that using a larger window context and relation modules are mostly complementary.

When more residual blocks are used and the head network becomes deeper (i), accuracy no longer increases. While, accuracy is continually improved when more relation modules are used (j).

The comparison indicates that the relation module is effective and the effect is beyond increasing network capacity.

Complexity In each relation module, $d_f = 1024, d_k = 64, d_g = 64$. When $N_r = 16$, a module has about 3 million parameters and 1.2 billion FLOPs, as from Eq. (7) and (8). The computation overhead is relatively small, compared to the complexity of whole detection networks as shown in Table. 5 (less than 2% for faster RCNN [38] / DCN [10] and about 8% for FPN [32]).

5.2. Relation for Duplicate Removal

All the experiments in this section use the detected objects of the Faster RCNN baseline $2fc$ head in Table 1 (top row, 29.6 mAP after NMS) for training and inference of our approach in Section 4.3.

In our approach, the relation module parameters are set

[5]Each residual branch in a block has three 1024-d fc layers to have similar complexity as an object relation module. The residual blocks are inserted at the same positions as our object relation modules.

backbone	test set	mAP	mAP_{50}	mAP_{75}	#. params	FLOPS
faster RCNN [38]	*minival*	32.2→34.7→**35.2**	52.9→55.3→**55.8**	34.2→37.2→**38.2**	58.3M→64.3M→64.6M	122.2B→124.6B→124.9B
	test-dev	32.7→35.2→**35.4**	53.6→**56.2**→56.1	34.7→37.8→**38.5**		
FPN [32]	*minival*	36.8→38.1→**38.8**	57.8→59.5→**60.3**	40.7→41.8→**42.9**	56.4M→62.4M→62.8M	145.8B→157.8B→158.2B
	test-dev	37.2→38.3→**38.9**	58.2→59.9→**60.5**	41.4→42.3→**43.3**		
DCN [10]	*minival*	37.5→38.1→**38.5**	57.3→57.8→**57.8**	41.0→41.3→**42.0**	60.5M→66.5M→66.8M	125.0B→127.4B→127.7B
	test-dev	38.1→38.8→**39.0**	58.1→**58.7**→58.6	41.6→42.4→**42.9**		

Table 5. Improvement (2fc head+SoftNMS [4], 2fc+RM head+SoftNMS and 2fc+RM head+e2e from left to right connected by →) in state-of-the-art systems on COCO *minival* and *test-dev*. Online hard example mining (OHEM) [40] is adopted. Also note that the strong SoftNMS method ($\sigma = 0.6$) is used for duplicate removal in non-e2e approaches.

as $d_f = 128, d_k = 64, d_g = 64, N_r = 16, N = 100$. Using larger values no longer increases accuracy. The duplicate removal network has 0.33 million parameters and about 0.3 billion FLOPs. This overhead is small, about 1% in both model size and computation compared to a faster RCNN baseline network with ResNet-50.

Table 3 investigates the effects of different input features to the relation module (Figure 3 (b)). Using $\eta = 0.5$, our approach improves the mAP to 30.3. When the rank feature is not used, mAP drops to 26.6. When the class score s_0 replaces the rank in a similar way (the score is embedded to 128-d), mAP drops to 28.3. When 1024-d appearance feature is not used, mAP slightly drops to 29.9. These results suggest that rank feature is most crucial for final accuracy.

When geometric feature is not used, mAP drops to 28.1. When it is used by *unary* method as mentioned in Section 3 and in Table 1 (a), mAP drops to 28.2. These results verify the effectiveness of our usage of geometric weight Eq. (5).

Comparison to NMS Table 4 compares our method with NMS method and its better variant SoftNMS [4], which is also the state-of-the-art method for duplicate removal.

Note that all three methods have a single parameter of similar role of controlling the localization accuracy: the IoU threshold N_t in NMS, the normalizing parameter σ in Soft-NMS [4], and the ground truth label criteria parameter η in ours. Varying these parameters changes accuracy under different localization metrics. However, it is unclear how to set the optimal parameters for NMS methods, other than trial-and-error. Our approach is easy to interpret because the parameter η directly specify the requirement on localization accuracy. It performs best for mAP_{50} when $\eta = 0.5$, for mAP_{75} when $\eta = 0.75$, and for mAP when $\eta \in [0.5, 0.9]$.

Our final mAP accuracy is better than NMS and Soft-NMS, establishing the new state-of-the-art. In the following end-to-end experiments, $\eta \in [0.5, 0.9]$ is used.

5.3. End-to-End Object Detection

The last row in Table 4 compares the end-to-end learning with separate training of instance recognition and duplicate removal. The end-to-end learning improves the accuracy by +0.5 mAP.

Finally, we investigate our approach on some stronger backbone networks, i.e., ResNet-101 [25] and better detection architectures, i.e., FPN [32] and DCN [10] in Table 5. Using faster RCNN with ResNet-101, by replacing the 2fc head with 2fc+RM head in Table 1 (default parameters), our approach improves by 2.5 mAP on COCO *minival*. Further using duplicate removal network with end2end training, the accuracy improves further by 0.5 mAP. The improvement on COCO *test-dev* is similar. On stronger baselines, e.g., DCN [10] and FPN [32], we also have moderate improvements on accuracy by both feature enhanced network and duplicate removal with end2end training. Also note that our implementation of baseline networks has higher accuracy than that in original works (38.1 versus 33.1 [10], 37.2 versus 36.2 [32]).

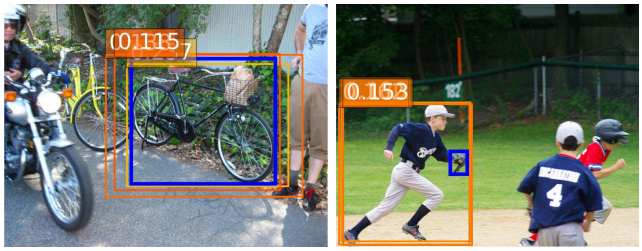


Figure 4. Representative examples with high relation weights in Eq. (3). The reference object n is blue. The other objects contributing a high weight (shown on the top-left) are yellow.

6. Conclusions

The comprehensive ablation experiments suggest that the relation modules have learnt information between objects that is missing when learning is performed on individual objects. Nevertheless, it is not clear what is learnt in the relation module, especially when multiple ones are stacked.

Towards understanding, we investigate the (only) relation module in the $\{r_1, r_2\} = \{1, 0\}$ head in Table 1(c). Figure 4 show some representative examples with high relation weights. The left example suggests that several objects overlapping on the same ground truth (bicycle) contribute to the centering object. The right example suggests that the person contributes to the glove. While these examples are intuitive, our understanding of how relation module works is preliminary and left as future work.

References

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

[2] 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*, pages 2874–2883, 2016.

[3] 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.

[4] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. Soft-nms–improving object detection with one line of code. In *CVPR*, 2017.

[5] D. Britz, A. Goldie, T. Luong, and Q. Le. Massive exploration of neural machine translation architectures. *arXiv preprint arXiv:1703.03906*, 2017.

[6] X. Chen and A. Gupta. Spatial memory for context reasoning in object detection. In *ICCV*, 2017.

[7] M. J. Choi, A. Torralba, and A. S. Willsky. A tree-based context model for object recognition. *TPAMI*, 34(2):240–252, Feb 2012.

[8] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2016.

[9] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016.

[10] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017.

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

[12] 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.

[13] Y. Duan, M. Andrychowicz, B. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. *arXiv preprint arXiv:1703.07326*, 2017.

[14] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *IJCV*, 2010.

[15] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 2010.

[16] C. Galleguillos and S. Belongie. Context based object categorization: A critical survey. In *CVPR*, 2010.

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

[18] R. Girshick. Fast R-CNN. In *ICCV*, 2015.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[20] G. Gkioxari, R. Girshick, and J. Malik. Contextual action recognition with r* cnn. In *ICCV*, pages 1080–1088, 2015.

[21] G. Gkioxari, R. B. Girshick, P. Dollár, and K. He. Detecting and recognizing human-object interactions. *CoRR*, abs/1704.07333, 2017.

[22] S. Gupta, B. Hariharan, and J. Malik. Exploring person context and local scene context for object detection. *CoRR*, abs/1511.08177, 2015.

[23] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.

[24] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

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

[26] J. Hosang, R. Benenson, and B. Schiele. Learning non-maximum suppression. In *ICCV*, 2017.

[27] 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 preprint arXiv:1611.10012*, 2016.

[28] 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. *IJCV*, 123(1):32–73, 2017.

[29] J. Li, Y. Wei, X. Liang, J. Dong, T. Xu, J. Feng, and S. Yan. Attentive contexts for object detection. *arXiv preprint arXiv:1603.07415*, 2016.

[30] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. *arXiv preprint arXiv:1611.07709*, 2016.

[31] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *arXiv preprint arXiv: 1708.02002*, 2017.

[32] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[33] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

[34] 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.

[35] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[36] 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.

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

[38] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[39] 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.

[40] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016.

[41] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[42] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[43] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[44] R. Stewart, M. Andriluka, and A. Y. Ng. End-to-end people detection in crowded scenes. In *ICCV*, 2016.

[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[46] A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin. Context-based vision system for place and object recognition. In *ICCV*, 2003.

[47] Z. Tu. Auto-context and its application to high-level vision tasks. In *CVPR*, 2008.

[48] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 2013.

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[50] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[51] B. Yao and L. Fei-Fei. Recognizing human-object interactions in still images by modeling the mutual context of objects and human poses. *TPAMI*, 34(9):1691–1703, Sept 2012.

[52] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014.

[53] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2017.