


# [Re] Object Detection Meets Knowledge Graphs


Jarl Lemmens<sup>1, </sup>, Pavol Jancura<sup>1</sup>, Gijs Dubbelman<sup>1</sup>, Hala Elrofai<sup>1</sup>


<sup>1</sup>Eindhoven University of Technology, Eindhoven, Netherlands

## Edited by

Benoît Girard 

## Reviewed by

Hirak Kashyap 

Stéphane Herbin 

## Received

08 October 2021

## Published

31 March 2023

## DOI

10.5281/zenodo.7788556

**A replication of** Y. Fang, K. Kuan, J. Lin, C. Tan, and V. Chandrasekhar. "Object detection meets knowledge graphs." In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17. International Joint Conferences on Artificial Intelligence, 2017, pp. 1661–1667.

## Summary of reproducibility

### Scope of reproducibility

'Object Detection meets Knowledge Graphs' [1] by Fang et al. describes a framework which integrates external knowledge from knowledge graphs, or background knowledge, into object detection. They apply two different approaches to quantify background knowledge as semantic consistency. An existing object detection algorithm is re-optimized with this knowledge to get updated knowledge-aware detections. The authors of [1] claim that this framework can be applied to any existing object detection algorithm and that this approach can increase recall, while maintaining mean Average Precision (mAP). In this work, the framework is implemented and the experiments are conducted as described in [1], such that the claims can be validated.

## Methodology

The authors in [1] describe a framework where a frequency based approach and a knowledge graph based approach are used to determine semantic consistency. A knowledge-aware re-optimization function updates the detections of a baseline Machine Learning object detection algorithm. Both the baseline and its re-optimized correlates are evaluated on two publicly available benchmark datasets, namely PASCAL VOC 2007 and MS COCO 2014. The replication of the experiments was completed using the information in [1] and clarifications of the author, as no source code was available. The framework was implemented in PyTorch and evaluated on the same benchmark datasets.

## Results

We were able to successfully implement the framework from scratch as described in [1]. We have bench-marked the developed framework on two datasets and replicated the results of all matrices as described in [1]. The claim of the authors of [1] can not be confirmed for both described approaches. The results either showed an increase of recall at the cost of a decrease in mAP, or a maintained mAP, without an improvement in recall. Three different backbone models show similar behavior after re-optimization,

---

Copyright © 2021 J.L.A. Lemmens, P. Jancura, G. Dubbelman, H.B.H. Elrofai, released under a Creative Commons Attribution 4.0 International license.

Correspondence should be addressed to Jarl Lemmens ([j.l.a.lemmens@student.tue.nl](mailto:j.l.a.lemmens@student.tue.nl))

The authors have declared that no competing interests exist.

Code is available at <https://github.com/tue-mps/rescience-ijcai2017-230>.

Data is available at <https://zenodo.org/record/7385730.Y73csnbMJPY> – DOI 10.5281/zenodo.7385730.

Open peer review is available at <https://github.com/ReScience/submissions/issues/59>.

concluding that the knowledge-aware re-optimization does not benefit object detection algorithms.

### What was easy

The methodology was well described and easy to understand conceptually.

### What was difficult

There was no source code available, which made it difficult to understand some technicalities of the implementation. The authors failed to mention a number of crucial details and assumptions of this implementation in the paper, which are essential for reproducing the methodology without making fundamental assumptions.

### Communication with the authors

We contacted the authors to elaborate on missing details, however no contact was found with the contact-information on the paper. Fortunately, a different email address of Yuan Fang was found online, to which he did respond fast and with clear explanations, for which we would like to express our gratitude.

## 1 Introduction

Object detection is one of the key tasks in computer vision. The PASCAL Visual Object Classes (VOC) challenge is one of the most accepted benchmarks for object detection. This challenge reads: "Where are the instances of a particular object class in the image (if any)?" [2]. In other words, the goal is to find a set of locations, or bounding boxes, in an image and to classify each instance with a label from a predefined set of classes [3]. Most state-of-the-art methods for object detection, such as Faster R-CNN [4], only use the visual features that are present in the images, while ignoring the vast amount of background information that is available.

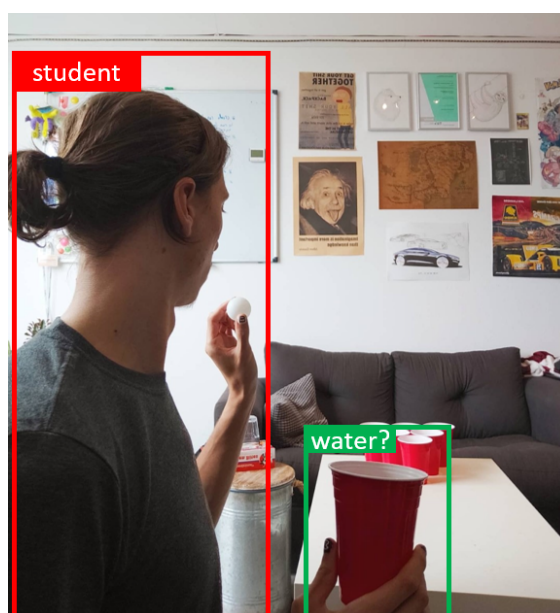
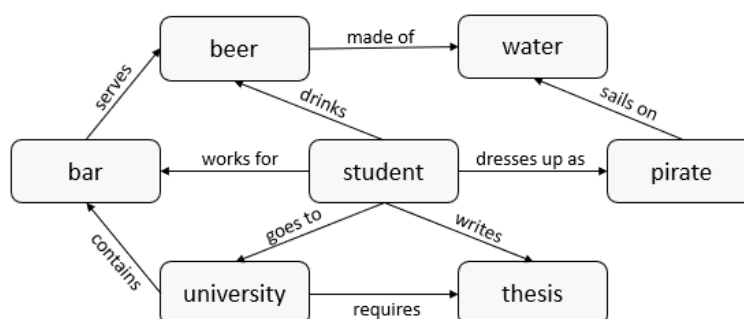


Figure 1. Toy example of an object detection.

Background information can be beneficial for classifying the detected objects [5]. For a human classifier, who has this background information at its disposal, it can be common sense that certain classes happen to co-occur more often than others. Figure 1 shows a scenario where a student is holding a drink. A human will classify the drink as a beer more likely than a water (assuming these are the only options), even though the liquid can not be identified directly. For machines, this ‘common sense’ is not self-evident. As a result, the machine is uncertain about the class of the drink and might even (wrongly) classify it as a water. One way to represent this background knowledge in a machine-readable way, is a knowledge graph. A knowledge graph nowadays has many different definitions [6]. In this paper it is defined as the original semantic network. This network is a knowledge base that represents semantic relations between concepts in a network such that the nodes represent the concepts and the edges represent the relationships between the nodes [7]. Figure 2 shows a toy example of such a knowledge graph, which would help identify the drink in Figure 1 as a beer rather than a water. It is expected that employing a large-scale knowledge graph as external background knowledge will improve the classification capabilities of the object detectors.



**Figure 2.** A toy knowledge graph modeling seven concepts as nodes (e.g., student and beer), as well as their relationships as edges (e.g., “student drinks beer”).

In this work we aim to replicate the findings of a knowledge-aware object detection framework, as described in ‘Object Detection meets Knowledge Graphs’ [1]. The proposed framework combines ‘traditional’ deep learning methods with the semantic consistency of a knowledge graph. The goal is to quantify and generalize the knowledge from the ConceptNet knowledge graph [8], and apply this semantic consistency to the object detection output of a Faster R-CNN [4] network to get an updated object prediction. Additionally, a frequency based method to determine semantic consistency directly from the datasets is applied. A custom re-optimization function is used to update the detections from a baseline object detection algorithm. The system is evaluated on the PASCAL VOC 2007 [2] and MS COCO 2014 [3] benchmark datasets.

## 2 Scope of reproducibility

The authors of [1] claim that any existing knowledge detection algorithm can be re-optimized by their proposed knowledge-aware detections. They propose two methods to extract semantic consistency between different classes from background information that are used to define the knowledge-aware detections. The first method is a frequency-based approach that uses information directly from the datasets and the second method is based on the information from a knowledge graph.

We aim to reproduce the claims that integrating background knowledge to conventional deep learning object detection algorithms will improve the recall while maintaining the same level of mean Average Precision (mAP).

## 3 Methodology

### 3.1 Model descriptions

The authors of [1] used a Caffe implementation of the Faster R-CNN network with a VGG-16 backbone pretrained on ImageNet.<sup>1</sup> Our work utilizes the readily available PyTorch implementation of the Faster R-CNN network with a pretrained ResNet50 backbone.<sup>2</sup> The model was trained separately on each dataset for evaluation on the specific dataset. Additionally, a VGG-16 backbone as well as a ResNet18 backbone were implemented and trained for supplementary experiments.

### 3.2 Datasets

**VOC** – The method is tested on two widely accepted benchmark datasets for object detection. The first is the PASCAL VOC 2007 dataset [2]. The VOC dataset consists of 10k images that are divided into 2.5k training set, 2.5k validation set and 5k test set.<sup>3</sup> The data is annotated for 20 different classes. The test set contains a number of objects that are annotated as 'difficult to detect'. For the challenge evaluation such objects are discarded, although no penalty is incurred for detecting them [2].

**COCO** – The second benchmark is the Microsoft COCO dataset [3]. COCO is an annual challenge that at times updates the dataset. For this approach, the 2014 dataset is used. This dataset consists of 165k images split into 83k for training, 41k for validation and 41k for testing.<sup>4</sup> For this (and the author's) approach, the training and validation set are combined for training, except for a subset of 5k images, that is divided again into 1k for validation and 4k for offline testing. The ground truth of the test set is not publicly available, thus evaluation is limited to the offline test set. COCO consists of 91 different classes, however 11 classes were later removed from annotation, which leaves 80 effective classes. This results in some images in the set being 'empty'. These images are ignored during training/inference. To align with the indexing of annotation, the model maintains the 91 classes, but the 11 redundant classes are ignored during evaluation. The 20 classes in the VOC dataset are a subset of the COCO classes.

### 3.3 Metrics

The main performance metrics that are used are recall@100 and mean Average Precision (mAP)@100. As stated in the COCO evaluation description, "all metrics are computed allowing for at most 100 top-scoring detections per image (across all categories)".<sup>5</sup> Both recall and AP are calculated per class and averaged at the end. The recall is determined per class according to

$$recall = \frac{TP}{TP + FN}, \quad (1)$$

where  $TP$  is the number of true positives, and  $FN$  is the number of false negatives. To simplify the implementation, the number of FN is not directly determined, but the sum of TP and FN is equal to the number of ground truth objects.

The AP is a precision-recall curve metric that calculates the weighted mean of precisions achieved at each recall threshold.<sup>6</sup> The cumulative number of true-positives (TP) and

<sup>1</sup><https://github.com/rbgirshick/py-faster-rcnn>

<sup>2</sup><https://pytorch.org/vision/stable/models.html>

<sup>3</sup><http://host.robots.ox.ac.uk/pascal/VOC/voc2007/index.html>

<sup>4</sup><https://cocodataset.org/download>

<sup>5</sup><https://cocodataset.org/detection-eval>

<sup>6</sup><https://blog.roboflow.com/mean-average-precision/>

false-positives (FP) are determined for the whole test set per class, in descending order of the confidence score. Subsequently, the recall is segmented into 101 equal parts, and for each segment the interpolated precision is determined, which is the maximum precision value for a given recall. A detection is considered to be a TP when the Intersection over Union (IoU) with the ground truth (of the correct class) is at least 0.5 for the VOC dataset. For the COCO dataset the mAP is calculated for different IoU thresholds  $\{0.50, 0.55, \dots, 0.95\}$  and then averaged. Besides recall@100, the recall@10 is determined for the COCO dataset, as well as the recall@100 specifically for small, medium and large objects.

### 3.4 Notation

Consider a set of pre-defined labels  $\mathcal{L} = \{1, 2, \dots, L\}$ . Let an existing object detection algorithm output a set of bounding boxes  $\mathcal{B} = \{1, 2, \dots, B\}$  for each image. Each bounding box  $b \in \mathcal{B}$  is assigned a label  $\ell \in \mathcal{L}$  with probability  $p(\ell|b)$ . Per image, these probabilities can be encoded by a matrix  $P$  of size  $B \times L$ . The goal is to construct a knowledge-aware matrix  $\hat{P}$  based on the semantic consistency knowledge. A bounding box is potentially assigned a new label  $\hat{\ell} = \operatorname{argmax}_{\ell}(\hat{P}_{b,\ell})$ .

### 3.5 Semantic consistency

To quantify background knowledge that is fundamentally symbolical and logical, it is proposed to determine a numerical degree of semantic consistency for each pair of concepts. The more likely that two concepts appear in the same image, the higher the degree of semantic consistency.

The matrix  $S$ , of size  $L \times L$ , is defined such that  $S_{\ell,\ell'}$  captures the degree of semantic consistency between the two concepts  $\ell$  and  $\ell'$ ,  $\forall(\ell, \ell') \in \mathcal{L}^2$ . The matrix is symmetrical, and the diagonal contains the self-consistency of a concept (a concept can occur more than once in the same image).

Two methods to determine semantic consistency are applied, in the same manner as the original paper.

**Frequency based method** – The simple way to compute  $S$  is to use the frequency of co-occurrences for each pair of concepts. In this work, both the VOC and COCO datasets are used to identify the co-occurrences. The semantic consistency is calculated, based on point-wise mutual information [9] using the following equation.

$$S_{\ell,\ell'} = \max\left(\log \frac{n(\ell, \ell') N}{n(\ell)n(\ell')}, 0\right) \quad (2)$$

Here  $n(\ell, \ell')$  denotes the frequency of co-occurrences for concepts  $\ell$  and  $\ell'$ ,  $n(\ell)$  denotes the frequency of  $\ell$  and  $N$  is the total number of instances in the background data.

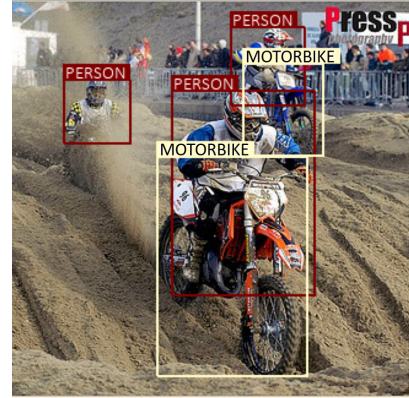
To determine the frequency of co-occurrences, the number of unique combinations of objects are counted for each image in both datasets. Then the results are summed for all images, before applying equation 2. The co-occurrences are given by the multiplication of  $n(\ell)$  and  $n(\ell')$ . For self-occurrences the ‘handshake’ equation

$$n(\ell, \ell) = n(\ell) \times \frac{n(\ell) - 1}{2} \quad (3)$$

is used. Figure 3 and Table 1 show a sample image to provide an example on how the (co-)occurrences are counted.

**Table 1.** Frequency of (co)-occurrences as counted from Figure 3

$n(\textit{person})$	3
$n(\textit{bike})$	2
$n(\textit{person}, \textit{bike})$	6
$n(\textit{bike}, \textit{person})$	6
$n(\textit{person}, \textit{person})$	3
$n(\textit{bike}, \textit{bike})$	1
$N$	5

**Figure 3.** Annotated VOC image

**Knowledge Graph based** – A more generalizing method to determine semantic consistency is to use a large-scale knowledge graph. A knowledge graph can also capture one or multiple chains of relations between concepts that are not directly connected to each other, which will increase robustness. To quantify these relations as semantic consistency, the ‘random walk with restart’ (RWR) approach is utilized [10]. From a starting node/concept, the algorithm ‘walks’ through the graph via neighbouring nodes. After every step there is a chance to teleport back to the starting node, to prevent from getting stuck in small localities. The probabilities of reaching certain nodes, eventually converge to a steady state. Equation 4 shows the iterative implementation of this approach.

$$\vec{r} \leftarrow (1 - c)\widetilde{W}\vec{r} + c\vec{e} \quad (4)$$

Here  $\vec{r}$  is the relatedness vector with respect to the starting node/seed  $\vec{e}$ .  $\widetilde{W}$  is a normalized weighted matrix that represents the weighted knowledge graph and  $c$  is the restart probability constant.

For this approach, a readily available implementation of the RWR algorithm is utilized. [11] As knowledge graph, the MIT ConceptNet version 5.7 is employed.<sup>7</sup> A list of assertions (two concepts and a relation between them) is filtered to only include the English segment. Also all negative relations (NotDesires, NotHasProperty, NotCapableOf, NotUsedFor, Antonym, DistinctFrom and ObstructedBy) and self-loops (when the relation directly connects a concept to the same concept) are ignored. The resulting graph has 1.16 million unique concepts and 3.35 million relations. To conform to the requirements of the RWR implementation, the graph is converted from concept strings to integers, using Pandas Dataframe structure. Note that the ConceptNet relations are directed, and when a bi-directional relation exists, these are added separately in the assertions list. Each relation comes with a weight corresponding to the significance of the relation, which is included in the RWR. The relatedness vector is computed for all 20 and 91 classes of the VOC and COCO dataset respectively. The matrix values of the 11 redundant COCO classes are set to 0.

### 3.6 Re-optimization

The re-optimization function is a result of optimizing the cost function given in equation 5.

<sup>7</sup><https://github.com/commonsense/conceptnet5/wiki/Downloads>

$$\begin{aligned}
E(\widehat{P}) = & (1 - \epsilon) \sum_{b=1}^B \sum_{\substack{b'=1 \\ b' \neq b}}^B \sum_{\ell=1}^L \sum_{\ell'=1}^L S_{\ell, \ell'} \left( \widehat{P}_{b, \ell} - \widehat{P}_{b', \ell'} \right)^2 \\
& + \epsilon \sum_{b=1}^B \sum_{\ell=1}^L B \|S_{\ell, *}\|_1 \left( \widehat{P}_{b, \ell} - P_{b, \ell} \right)^2
\end{aligned} \tag{5}$$

Here, the first term captures the constraint on semantic consistency that for two different bounding boxes  $b$  and  $b'$  in the same image,  $P_{b, \ell}$  and  $P_{b', \ell'}$  should not be too different for a large value of  $S_{\ell, \ell'}$ . The second term captures the constraint such that the knowledge-aware detections do not diverge too much from the original detections. The  $B \|S_{\ell, *}\|_1$  coefficient balances both terms, because of the different number of summations. Optimizing this cost function 5 results in the re-optimization function given in equation 6 [1].

$$\widehat{P}_{b, \ell}^{(i)} = (1 - \epsilon) \frac{\sum_{b'=1, b' \neq b}^B \sum_{\ell'=1}^L S_{\ell, \ell'} \widehat{P}_{b', \ell'}^{(i-1)}}{\sum_{b'=1, b' \neq b}^B \sum_{\ell'=1}^L S_{\ell, \ell'}} + \epsilon P_{b, \ell} \tag{6}$$

For any arbitrary initialization  $\widehat{P}_{b, \ell}^{(0)}$ ,  $\widehat{P}_{b, \ell}^{(i)}$  converges to the same solution eventually (usually within 30 iterations). To speed up the computation, an approximation is applied where the  $B_k$  nearest bounding boxes and  $L_k$  nearest concepts are considered. This means that for a bounding box  $b$ , the nearest  $B_k$  boxes to  $b$  (shortest distance center-to-center),  $b'$  are considered. For a concept  $\ell$ , only the top  $L_k$  concepts  $\ell'$  with the largest semantic consistency to  $\ell$  are considered. To increase the efficiency for computing equation 6, matrix multiplication is used. Using the  $L_k$  approximation, a sparse consistency matrix can be constructed.

### 3.7 Hyper-parameters

All hyper-parameters are kept identical as described in the original paper. The models (for VOC and for COCO), are trained using Stochastic Gradient Decent with a momentum of 0.9, a weight-decay of 5e-4 and a batch-size of 2. A learning rate of 1e-3 is used for the first 50k/350k iterations, followed by 1e-4 for another 10k/140k iterations on VOC/COCO respectively.

The default parameters of the Faster R-CNN model are kept, except the (maximum) number of detections per image is set to 500, and the score threshold of the bounding boxes is decreased to 1e-5. The random walk restarting probability  $C$  is set to 0.15. The hyper-parameter  $\epsilon$  in equation 6 is chosen between  $\{0.1, 0.25, 0.5, 0.75, 0.9\}$  on the validation sets. After contact with the author,  $\epsilon$  was set to 0.9 for the VOC dataset and 0.75 for the COCO dataset. The updates of equation 6 are performed for 10 iterations, and  $B_k$  and  $L_k$  are both set to 5.

### 3.8 Experimental setup & code

Figure 4 shows a flowchart of the implementation during inference. A test image is fed to the Faster R-CNN network, where the ResNet-50 backbone produces a set of region proposal bounding boxes (1000 boxes by default, the objectness score threshold is 0.0) that are classified such that each class gets a probability score  $P_{b, \ell}$ . The bounding box proposals are updated by a regression layer such that each class gets a separate bounding box with the corresponding class score. During post-processing the bounding boxes are filtered on a score-threshold (1e-5 in this approach). Double detections are filtered out by applying a class-wise Non-Maxima-Suppression (NMS). Next, a maximum of 500 top scoring boxes are kept. It is important to note that at this point for each bounding box only the highest score probability remains. For the baseline approach, the top 100

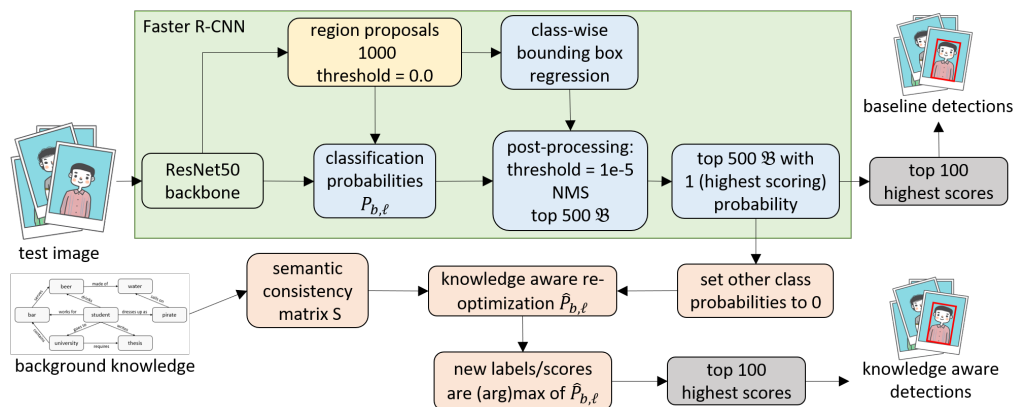


Figure 4. Flowchart of the knowledge aware re-optimization during inference.

highest scores are selected and used for evaluation. For the knowledge aware approach, the probabilities of all classes other than the highest scoring class of that bounding box are set to 0 initially. Then the re-optimization function 6 is applied. This was not mentioned in the published paper, however it is crucial for implementation. For each detection, the maximum of  $\hat{P}_{b,\ell}$  will be the (updated) score for that bounding box, and there is a chance that the label will be updated as well. Again the top 100 highest scores are considered for evaluation.

Three different variants of the semantic consistency matrix are constructed and evaluated. First, the frequency based method is applied on both datasets. To conform to the published results, we keep the same naming, so this approach is called KF-All. A second approach uses only 250 images of each dataset to show the relevance of background quality. We call this approach KF-500. The third approach utilizes the ConceptNet knowledge graph and is called KG-CNet57, which refers to as the version 5.7 of the graph. Additionally we check the difference between this version and an older graph (version 5.5), which we call KG-CNet55. The baseline Faster R-CNN results are called FRCNN. As no source code was available from [1], the methodology was implemented from scratch based on the descriptions in the paper and contact with the author.

## 4 Results

### 4.1 VOC

The obtained results support the claims presented in [1], only in one specific case. Table 2 shows the originally published results as well as our results on the VOC dataset for the different backbone models, namely ResNet-50, ResNet-18 and VGG-16. The first observation is that for the ResNet-50 frequency based approaches, the mAP is maintained at 70.1, which is in line with the claims of [1], as discussed shortly. However, the recall is only increased by 0.1 for the KF-All, which is not comparable to the published benefits in [1]. For the knowledge graph based approaches, both mAP and recall are decreased with 0.3 using the re-optimization process. The ResNet-18 results show a maintenance or decrease for both mAP and recall similar to the results of ResNet-50 for both the frequency based and the knowledge graph based approaches. This shows that there is hardly any difference in the effect of re-optimization between a relatively 'worse' or 'better' backbone. The VGG-16 results again show similar behavior of decrease in mAP, while in the best case maintaining recall. The difference between published results and our VGG-16 baseline results can be attributed to the re-implementation of the original Caffe model to the PyTorch environment. This is done to the best of our knowledge, but there might be some operational discrepancies.



**Table 2.** Comparison of the baseline results and the knowledge-aware variants on the PASCAL VOC 2007 test set for various backbone models.

	mAP @100	Recall@100 by concepts																				
Published		all	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
FRCNN	66.5	81.9	76.1	89.0	74.3	73.4	64.6	89.7	85.8	90.5	69.0	88.9	85.4	91.6	92.0	85.2	82.4	60.8	83.1	89.1	84.4	82.1
KF-500	66.6	83.8	80.0	91.7	79.1	76.0	67.0	89.7	88.8	92.5	69.7	92.6	85.9	90.8	94.0	86.8	82.0	59.6	87.2	90.0	89.7	82.8
KF-All	66.5	84.6	80.7	93.5	79.1	76.0	67.6	90.1	88.8	93.6	68.1	93.0	86.9	94.1	93.1	89.5	83.1	65.4	88.0	89.1	90.1	81.8
KG-CNet	66.6	85.0	80.4	92.3	78.6	76.0	67.6	90.1	89.1	92.2	74.2	93.0	86.4	93.0	92.2	88.6	87.7	66.9	87.6	90.4	89.7	83.4
ResNet-50																						
FRCNN	70.1	92.7	92.3	94.7	89.8	88.6	81.9	96.2	96.2	97.5	85.6	98.8	94.7	98.6	96.8	93.8	93.3	77.3	93.8	97.1	94.7	91.6
KF-500	70.1	92.3	93.3	95.0	90.6	85.6	80.8	96.2	96.5	97.5	84.4	99.2	94.2	98.8	96.6	94.8	92.4	76.3	93.0	96.2	95.4	90.3
KF-All	70.1	92.8	93.0	95.3	90.6	89.0	80.8	96.2	96.4	97.2	84.5	99.2	94.7	98.8	97.1	95.7	92.6	76.3	94.2	96.7	95.7	91.9
KG-CNet57	69.8	92.4	93.3	94.7	90.0	88.6	78.7	96.7	96.4	96.9	83.6	99.2	93.7	98.4	95.7	94.8	94.0	77.1	92.1	97.5	95.4	91.9
KG-CNet55	69.8	92.4	93.3	94.7	90.0	88.6	78.7	96.7	96.4	96.9	83.6	99.2	93.7	98.4	95.7	94.8	94.0	77.1	92.1	97.5	95.4	91.9
ResNet-18																						
FRCNN	63.4	90.5	98.7	93.8	84.7	84.8	74.2	95.8	93.8	96.4	83.9	97.5	91.7	96.5	96.3	92.6	92.3	73.5	91.3	97.5	94.3	90.6
KF-500	63.3	90.2	88.4	93.2	85.8	82.9	74.4	96.2	93.8	96.9	82.3	97.5	90.8	96.7	95.7	92.9	91.1	72.9	90.9	96.7	95.7	89.9
KF-All	63.4	90.5	88.1	93.2	85.6	84.8	74.4	96.2	93.8	96.9	83.1	97.5	92.7	96.9	95.7	93.2	91.3	73.1	91.7	96.2	95.7	89.9
KG-CNet57	63.2	90.2	87.0	93.2	85.0	85.2	73.8	95.8	94.0	96.6	83.3	97.1	90.3	97.1	94.0	92.9	92.9	73.1	90.5	96.7	96.5	88.6
KG-CNet55	63.2	90.2	87.0	93.2	85.0	85.2	73.8	95.8	94.0	96.6	83.3	97.1	90.3	97.1	94.0	92.9	92.9	73.1	90.5	96.7	96.5	88.6
VGG-16																						
FRCNN	62.7	93.0	93.0	96.7	89.8	89.4	80.2	95.8	96.1	96.9	88.1	98.8	93.7	98.2	96.3	95.7	93.7	80.4	93.0	97.5	95.0	91.6
KF-500	62.5	92.8	94.7	96.4	89.8	88.2	79.3	95.8	96.8	97.5	86.2	98.4	91.7	97.3	96.0	96.0	93.0	80.6	93.4	97.1	96.1	91.2
KF-All	62.6	93.0	94.0	96.4	90.6	89.4	79.5	95.8	96.8	97.5	86.8	98.8	92.2	97.5	96.3	96.3	93.1	80.6	93.0	97.1	96.1	91.9
KG-CNet57	62.2	92.5	93.0	96.1	88.2	89.0	78.0	95.8	96.3	97.2	87.2	98.0	92.2	96.7	94.5	95.7	94.3	80.2	93.0	97.1	95.7	90.9
KG-CNet55	62.2	92.5	93.0	96.1	88.2	89.0	78.0	95.8	96.3	97.2	87.2	98.0	92.2	96.7	94.5	95.7	94.3	80.2	93.0	97.1	95.7	90.9

## 4.2 COCO

Similar to the VOC dataset, Table 3 shows the originally published results as well as our results on the COCO dataset for different backbones, namely ResNet-50 and VGG-16. The KF-All based approach shows a decrease in mAP of 0.2 for both backbones, but an increase in recall of 0.4 and 0.3 respectively. The authors of [1] claim that the knowledge graph approaches perform better on the COCO dataset compared to the VOC dataset. The results show a larger increase in recall (+0.7 compared to -0.3), but also a larger decrease in mAP (-0.7 compared to -0.3), so this claim can be argued as well. Additional to the recall, the COCO dataset also looks at the recall per area of the objects, as well as the recall@10. For the published results, the recall@10 shows improvements for all different approaches as well, whereas for our results, the recall@10 stays more or less the same. This is an indication that the knowledge aware approach mostly affects the lower scoring detections. To test this, an extra set of experiments were performed. It can be concluded that the KF-All approach indeed performs better than the KF-500 approach for all backbones and datasets. It should be noted that the results for the two different versions of the ConceptNet knowledge graph show no difference at all.

**Table 3.** Comparison of the baseline results and the knowledge-aware variants on the MS COCO 2014 test split. Published results are marked with “\*”.

	mAP @100	recall		recall@100 by area		
		@100	@10	small	medium	large
Published						
FRCNN	24.5	35.9	35.2	14.2	41.5	55.6
KF-500	24.4	37.1	35.6	14.3	42.8	57.3
KF-All	24.5	37.9	36.2	14.6	43.9	58.6
KG-CNet	24.5	38.9	36.6	14.4	45.2	60
ResNet-50						
FRCNN	27.9	47.2	32.3	30.4	50.4	59.6
KF-500	27.5	47.2	32.3	30.0	50.4	59.7
KF-All	27.7	47.6	32.3	30.5	50.9	60.2
KG-CNet57	27.2	47.9	32.1	29.9	51.0	61.1
KG-CNet55	27.2	47.9	32.1	29.9	51.0	61.1
VGG-16						
FRCNN	24.3	44.1	29.0	27.8	46.7	56.5
KF-500	24.0	44.0	29.0	27.8	46.8	56.1
KF-All	24.1	44.4	29.0	28.0	47.2	56.6
KG-CNet57	23.7	44.7	28.8	27.6	47.5	57.4
KG-CNet55	23.7	44.7	28.8	27.6	47.5	57.4

## 4.3 Results beyond the paper

To show the effects of the re-optimization process, two more sets of experiments were performed on the VOC dataset. Firstly, the effects of the bounding box score threshold were experimented. A set of 6 different thresholds were tested in the range of  $1e-5$ , as used in the published method, and 0.05, which is the default value for Faster R-CNN implementations. Even though the maximum detections per image is kept at 500, only a handful of detections (averaged at 6.2 per image) surpass this threshold value. This higher threshold experiment can show the effects of the re-optimization from a more practical point of view.

Figure 5 and 6 show the resulting mAP@100 and recall@100 (averaged per class) for the different thresholds. Only one knowledge graph based (KG-CNet57) and one frequency based approach (KF-All) were tested, in addition to the baseline. The results show that for a practical threshold score the recall is not affected at all, meaning that there is no increase in True Positive detections. The decrease in mAP implies that the detections that changed label after re-optimization, on average, changed for the worse. This decrease is present for the 0.05 threshold, but largest at the 1e-2 mark.

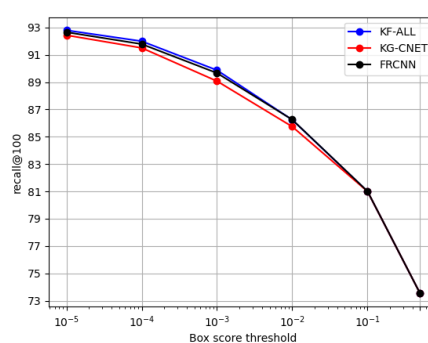
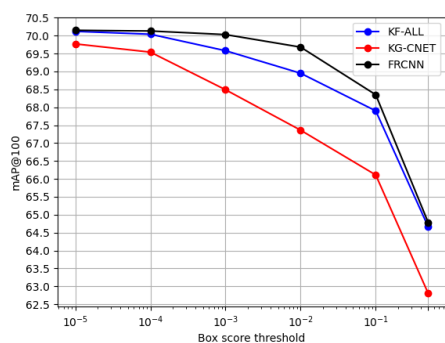


Figure 5. mAP@100 for various box score thresholds

Figure 6. recall@100 for various box score thresholds

The second set of experiments were performed to show the effect of the re-optimization parameter  $\epsilon$ . mAP@100 and recall@100 are again reported for the two re-optimization approaches for different values of  $\epsilon$ . A range of [0.05, 1.0] in steps of 0.05 was reported. The results are shown in Figure 7 and 8. It should be noted that an  $\epsilon$  of 1 is the same as using the FRCNN baseline. The results show that the chosen value of 0.9 is indeed optimal compared to the other values, but it also shows that by increasing the knowledge awareness part, the performance is affected negatively.

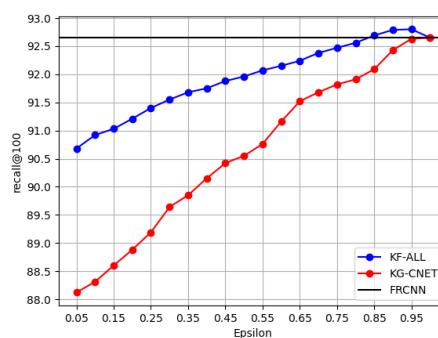
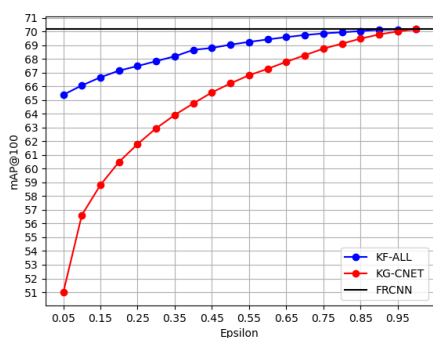


Figure 7. mAP@100 for various values of epsilon

Figure 8. recall@100 for various values of epsilon

## 5 Discussion

As the source codes were not published for the original paper, the methods had to be implemented from scratch. During implementation it became evident that some details were missing in the description of the methodology. Mainly, the lack of parameters to determine baseline results, the missing final hyper-parameter  $\epsilon$ , and the fact that the

non-highest-scoring class probabilities are set to 0. Without clarification on these details by the author of [1], the reproducibility of the paper would be based on our own assumptions. Especially the latter is a crucial, non-trivial implementation detail, since the methodology clearly describes the use of different class probabilities for each bounding box. As Faster R-CNN only outputs the one highest scoring probability per bounding box, it is ambiguous where the other probabilities come from. For simplification these probabilities were set to 0, an important fact that should have been mentioned in the original paper.

Based on the results presented in Tables 2 and 3, none of the re-optimization approaches confirms the claim of maintaining mAP while increasing recall on both the VOC as COCO dataset. A similar trend in re-optimization effects is shown for two completely different backbone models (ResNet-50 and VGG-16). Additionally, a simpler version of the used backbone model (ResNet-18) also shows no benefits for either a 'better' or 'worse' object detection model, so it cannot be claimed that a better model infers the semantic knowledge by itself.

It can be concluded that our reproduced results contradict the claims of the authors of the original paper [1]. Their claim stated that the semantic consistency in background knowledge can be applied to 'any' object detection model and increase performance. By failing to produce similar results, for various models, including the one used in [1], the claim cannot be reproduced and confirmed.

The results do show that the quality of background information matters, as the frequency based approach that uses all dataset images, consistently outperforms the same approach that uses only 500 images. The fact that there is no difference at all between two versions of the knowledge graphs shows that there might be more beneficial ways to extract the semantic consistency matrices from the graphs. Additional experiments were done to show the effects of the knowledge aware re-optimization on different box score thresholds. It can be concluded that for a more practical object detection situation with a threshold of 0.05 (default in object detection), there are no improvements in performance. An additional experiment was done to show the effect of hyperparameter  $\epsilon$  as this was not presented by the original paper.

## 5.1 What was easy

The methodology of the paper was written clearly, which made it conceptually easy to comprehend what was going on. It was easy to understand which datasets, networks and metrics were used for their experiments and their results gave a clear overview for us to compare our reproductions.

## 5.2 What was difficult

The paper was not supported by any available code, which meant that the reproduction had to be built from scratch. This was the main reason to choose for the PyTorch environment. Unfortunately there was no VGG16-backed Faster R-CNN directly available in this environment. Therefore the closest related model was chosen, which was the ResNet50-backed Faster R-CNN. After reviewing the results, another attempt was made to include the VGG-16 backed results as well, which was not a simple process. During implementation of the methodology it became apparent that some crucial details were missing from the paper. Mainly the fact that other class probabilities were set to 0 after extracting the highest scores from the model. Also the final used value of  $\epsilon$  was not mentioned and the lack of explanation of baseline parameters made it difficult to get to the correct implementation.

### 5.3 Communication with the authors

The paper mentioned the email addresses of 5 authors, out of which 2 were rejected. After contacting the others 3 times, there was still no response. By coincidence, a different email address of Yuan Fang was found online, to which he responded quickly and provided us with the missing details and other further explanations.

## References

1. Y. Fang, K. Kuan, J. Lin, C. Tan, and V. Chandrasekhar. "Object detection meets knowledge graphs." In: **Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, {IJCAI-17}**. International Joint Conferences on Artificial Intelligence, 2017, pp. 1661–1667.
2. M. Everingham, L. Van Gool, C. K. I Williams, J. Winn, A. Zisserman, M. Everingham, L. K. Van Gool Leuven, B. CKI Williams, J. Winn, and A. Zisserman. "The PASCAL Visual Object Classes (VOC) Challenge." In: **Int J Comput Vis** 88 (2010), pp. 303–338.
3. 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: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. Vol. 8693 LNCS. PART 5. Springer Verlag, May 2014, pp. 740–755.
4. S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." In: **IEEE Transactions on Pattern Analysis and Machine Intelligence** 39.6 (June 2017), pp. 1137–1149.
5. E. Krupka and N. Tishby. "Incorporating prior knowledge on features into learning." In: **Journal of Machine Learning Research**. Vol. 2. 2007, pp. 227–234.
6. L. Ehrlinger and W. Wöb. "Towards a definition of knowledge graphs." In: **CEUR Workshop Proceedings**. Vol. 1695. 2016.
7. J. F. Sowa. "Semantic Networks." In: **Encyclopedia of Artificial Intelligence** (1992).
8. R. Speer, J. Chin, and C. Havasi. "ConceptNet 5.5: An Open Multilingual Graph of General Knowledge." In: (Dec. 2016).
9. G. Bouma. "Normalized ( Pointwise ) Mutual Information in Collocation Extraction." In: **Proceedings of German Society for Computational Linguistics (GSCL 2009)** (2009), pp. 31–40.
10. H. Tong, C. Faloutsos, and J.-Y. Pan. **Fast Random Walk with Restart and Its Applications**. Tech. rep.
11. J. Jung. **Python Implementation for Random Walk with Restart (RWR)**. 2021.