Project Documentation

Study, Design, and Implement the Knowledge-aware Object Detection Symbolic Knowledge Injection Algorithm

Pelinsu Acar, Rubin Carkaxhia, and Calin Diaconu

Ethics in Artificial Intelligence
August 30, 2024

Abstract

Object detection is one of the most popular computer vision tasks, with solutions to the problem existing since before the spread of modern convolutional neural networks, or the newer transformer architecture. Applications can be found in a wide range of fields, from autonomous vehicles and surveillance, to medical imagining and wildlife tracking. The current project aims to implement and, potentially, improve a framework presented in a 2017 paper [1]. This framework is based on the notion of ”semantic consistency”, adding background information about the real world, and works by including external knowledge from a knowledge graph, alongside the image information. It reports an increase of 6.3 points in recall, compared to the bare-bone object detection solutions. In this regard, this documentation details the implementation process of the proposed methodology, the challenges we faced due to some drawbacks of the original paper and finally the results achieved with a comparison of original values presented by the authors of this framework.

1. Introduction

Object detection refers to the task of localizing and classifying, inside an image, all the objects that belong to one of the predetermined classes of interest. The localization is given through a bounding box, described through four parameters (the x and y coordinates of two of its opposite corners, or the coordinates of its center and the width and height, all relative to image dimensions), which also informs on the relative size of the object. Traditionally, these techniques only leverage information that shows up in the image under evaluation, without considering background information about the world, that could provide insight into the relationships between the identified objects, much like humans do.

This paper explores methods of quantifying and generalizing knowledge, as well as incorporating this knowledge to achieve "knowledge-aware object detection." By following the steps described in 'Object Detection meets Knowledge Graphs' [1], we aim to observe the impact of semantic knowledge on standard object detection methods. The authors of [1] argue that this knowledge-aware approach can be implemented into any existing object detection algorithm. Therefore, during our work, we experiment not only on Faster R-CNN [2], which was used by the original authors, but also on more novel detection algorithms such as YOLOv8 [3] and DETR [4]. The final results are evaluated on the PASCAL VOC 2007 [5] benchmark datasets to assess the impact of incorporating semantic knowledge into object detection models.

2. Background and Motivation

Humans naturally use contextual information to make sense of the world around them. For instance, in a scenario where a cat is climbing a tree (as shown in Figure 1), it is much more common for a human to associate a cat with climbing a tree than a bear. This contextual understanding allows humans to more easily categorize the animal in the image. However, traditional object detection models lack this kind of common sense, as they do not consider background information that could provide valuable insights into the relationships between objects.

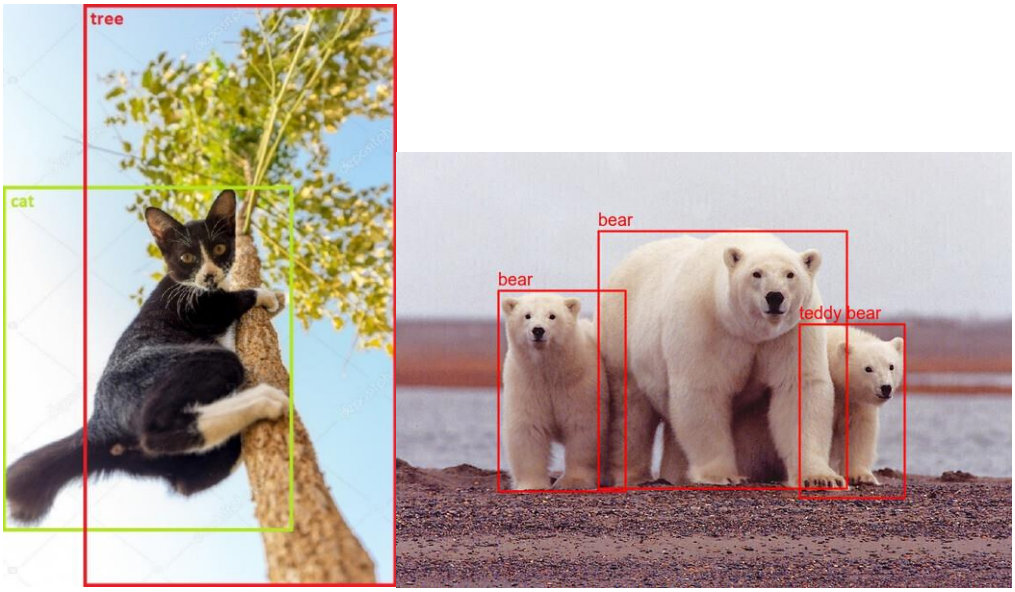

Figure 1: Object detection on images (Left: Detecting cat and tree, Right: Detecting bear)

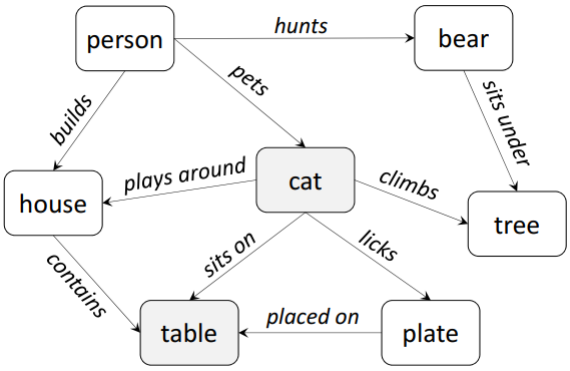

Figure 2: A toy knowledge graph modeling seven concepts as nodes (e.g., cat and tree), as well as their relationships as edges (e.g., “cat climbs tree”)

One way to provide such information to machines is through the use of knowledge graphs, which model semantic knowledge. In a knowledge graph, nodes represent real-world concepts, and edges represent the connections between these concepts. As it can be seen from the example in figure 2, that kind of a knowledge graph would provide insight about the relationship between cat, tree and bear. By employing a much larger knowledge graph, it is expected for our object detection models to have more common sense about the predicted objects and therefore to get better predictions [6]. The authors of 'Object Detection meets Knowledge Graphs' [1] propose methods for extracting semantic consistency between different classes using knowledge graph-based and frequency-based approaches. These methods aim to re-optimize object detection by incorporating semantic knowledge, ultimately leading to improved detection accuracy. In our work, we implement a custom re-optimization function to update the predictions using both knowledge graph-based and frequency-based approaches.

3.1 Related Work

In recent years, deep convolutional neural networks (CNNs) have become the standard for computer vision tasks, including image classification and object detection, due to their ability to learn high-level image features. Early models like R-CNN and its fast variant relied on precomputed region proposals for object localization, while later models such as Overfeat and Faster R-CNN [2] integrated CNNs for both object classification and localization, with Faster R-CNN introducing an efficient region proposal network. More recently, models like YOLOv8 [3] and DETR [4] have set new benchmarks in object detection. YOLOv8, the latest iteration of the "You Only Look Once" series, is known for its real-time detection capabilities and improved accuracy by using a fully convolutional architecture. DETR [4] (DEtection TRansformers) represents a shift towards transformer-based architectures, where object detection is treated as a direct set prediction problem, allowing for end-to-end optimization without the need for traditional region proposals. Beyond image data, there's a growing interest in utilizing external knowledge, such as texts and knowledge graphs, for tasks like image classification [7], visual relationship extraction [8], and visual question answering [9]. However, the application of external knowledge in object detection has been limited. Earlier approaches like conditional random fields [10] and co-occurrence statistics [11] aimed to incorporate semantic context but struggled with generalization. In contrast, knowledge graphs, which model real-world concepts and their relationships, offer a more generalizable approach. These graphs have gained prominence in various data-driven applications, including web search and social networks, and are constructed through methods ranging from human curation to automatic extraction [12]. Recent efforts have also begun to incorporate multimodal data, including images, into knowledge graphs, offering new possibilities for knowledge-aware object detection.

3.2 Model Descriptions

1. Faster R-CNN

- **Architecture**: Faster R-CNN [2] is a two-stage object detection model. It consists of a **Region Proposal Network (RPN)** and a **Fast R-CNN** detector.

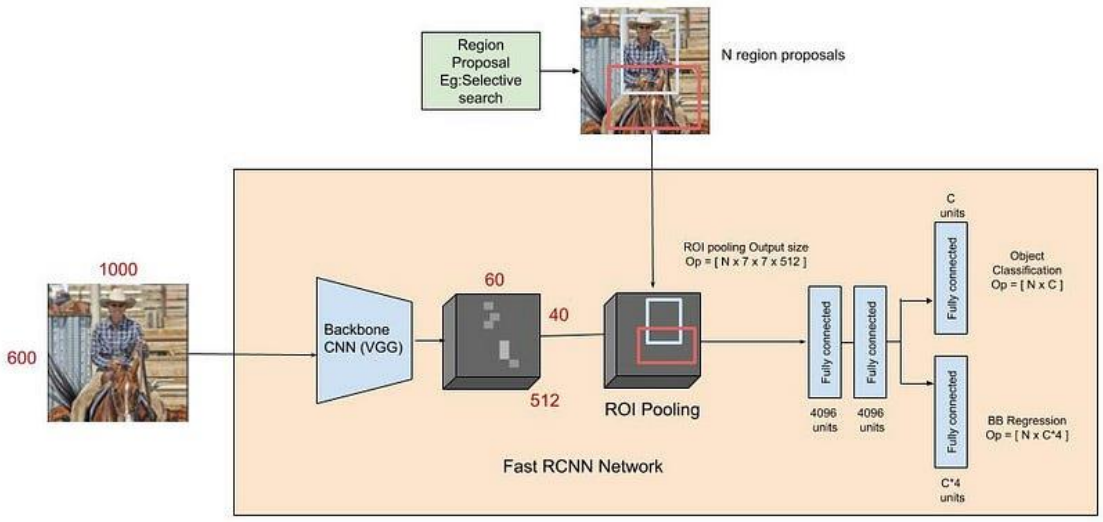

Figure 3: Overall Architecture of Faster R-CNN

- **Region Proposal Network (RPN)**: This network generates region proposals, which are potential bounding boxes where objects might be located. The RPN slides a small network over the feature map output by a CNN backbone (e.g., ResNet, VGG) and predicts objectness scores and bounding box coordinates.
- **Fast R-CNN**: The proposed regions are then fed into the Fast R-CNN, which classifies each region and refines the bounding box coordinates.
- **Anchor Boxes**: Faster R-CNN uses predefined anchor boxes of different scales and aspect ratios to detect objects of various sizes.
- **Feature Pyramid Network (FPN)**: An optional component that helps in detecting objects at different scales by creating feature maps at different resolutions.

2. YOLOv8

- **Architecture**: YOLOv8 [3] is a single-stage object detection model, which means it predicts bounding boxes and class probabilities directly from feature maps in one go.

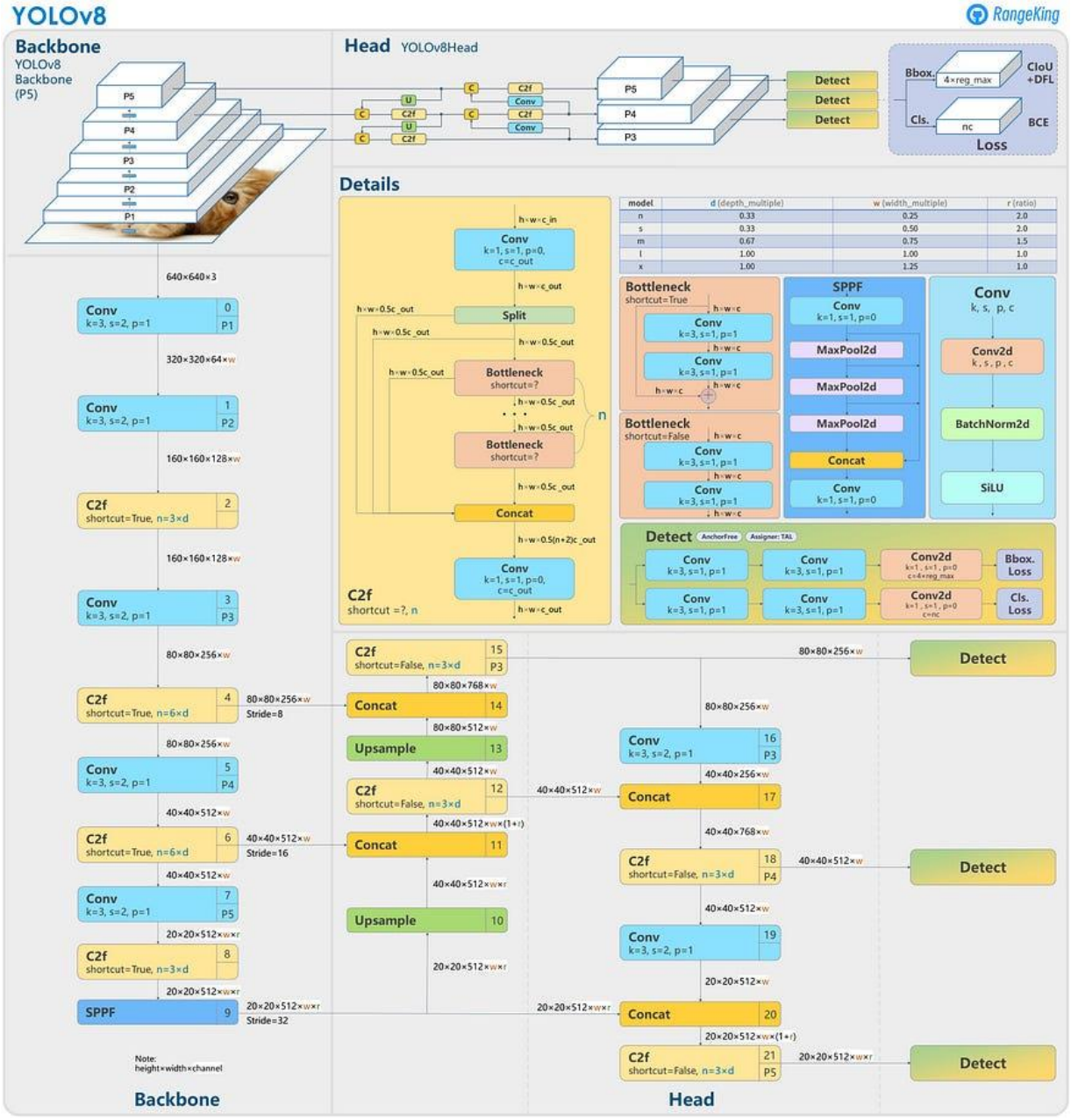

Figure 4: Overall Architecture of Yolov8

- **Backbone**: YOLOv8 uses a custom convolutional neural network as its backbone to extract features from the input image.
- **Head**: The model predicts bounding boxes, objectness scores, and class probabilities using a single convolutional head. Unlike its predecessors, YOLOv8 has improved efficiency and accuracy through innovations like better anchor-free detection and deeper, more efficient feature extraction layers.
- **Anchor-Free Detection**: YOLOv8 moves away from traditional anchor boxes, making the model faster and more flexible by directly predicting bounding box centers, width, and height.
- **Multi-Scale Detection**: YOLOv8 incorporates multiple feature maps of different resolutions to detect objects at various scales.

3. DETR (Detection Transformers)

- **Architecture**: DETR [4] represents a significant shift in object detection by using transformers instead of traditional convolutional methods.

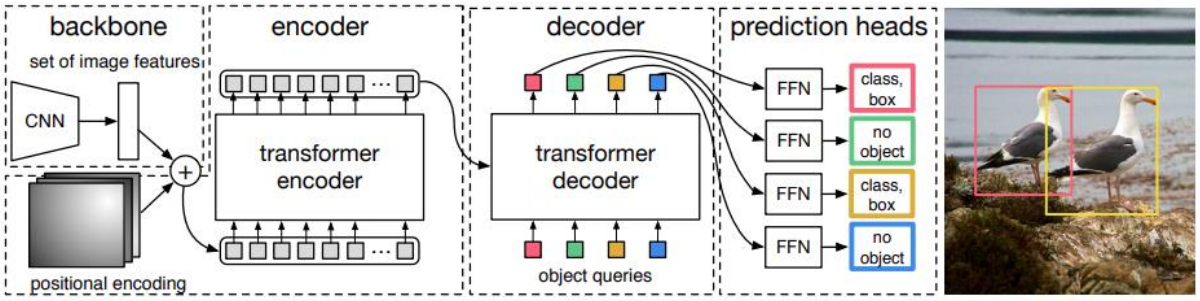

Figure 5: Overall Architecture of DETR

  - **Backbone**: Similar to other models, DETR uses a CNN (e.g., ResNet) to extract feature maps from the input image.
  - **Transformer Encoder-Decoder**: The extracted features are then passed through a transformer encoder-decoder architecture. The encoder processes the entire image contextually, while the decoder generates a fixed number of object queries, each predicting a bounding box and a class label.
  - **Set-Based Prediction**: DETR treats object detection as a set prediction problem, where the model outputs a fixed-size set of predictions and matches them to the ground truth via a bipartite matching loss.
  - **No Post-Processing**: Unlike traditional models, DETR does not require post-processing steps like Non-Maximum Suppression (NMS) because it directly outputs final predictions without overlapping.

Key Differences

- **Detection Paradigm**:
  - **Faster R-CNN**: Two-stage detector with separate region proposal and classification stages.
  - **YOLOv8**: Single-stage detector with fast, direct predictions.
  - **DETR**: Transformer-based set prediction model without region proposals or anchor boxes.
- **Anchor Boxes**:
  - **Faster R-CNN**: Uses predefined anchor boxes.
  - **YOLOv8**: Moves towards anchor-free detection.
  - **DETR**: No anchor boxes, uses transformers for direct bounding box prediction.
- **Training Complexity**:
  - **Faster R-CNN**: More complex with multiple stages and hyperparameters.
  - **YOLOv8**: Simplified, fast training with a focus on real-time detection.
  - **DETR**: Simple but requires more training epochs and computational power.
- **Speed vs. Accuracy**:
  - **Faster R-CNN**: Slower but more accurate.
  - **YOLOv8**: Balanced with high speed and good accuracy.
  - **DETR**: Accurate with better contextual understanding, but training is slower.

4. Methodology

4.1 Semantic Consistency

To quantify background knowledge, which is inherently symbolic and logical, a numerical measure of semantic consistency is proposed for each pair of concepts. The idea is that the more frequently two concepts are likely to appear together in the same image, the higher their semantic consistency score.

This relationship is captured in a matrix S of size L×L, where $S_{ℓ,ℓ'}$ represents the degree of semantic consistency between two concepts ℓ and ℓ', for all pairs (ℓ,ℓ') within the set L^2. The matrix is symmetrical, meaning that $S_{ℓ,ℓ'} = S_{ℓ',ℓ}$ and the diagonal entries represent the self-consistency of each concept, acknowledging that a concept can appear multiple times within the same image.

To calculate these semantic consistency values, two methods are employed, consistent with the approach taken in the original research.

Frequency-based knowledge

A straightforward method to compute the semantic consistency matrix S is by analyzing the frequency of co-occurrences for each pair of concepts. In this study, we experiment on the VOC datasets to find these co-occurrences. The semantic consistency between concepts is calculated using point-wise mutual information (PMI) [13], as described by the following equation:

$$S_{\ell,\ell'} = \max\left(\log \frac{n\,(\ell,\ell')\,N}{n(\ell)n\,(\ell')}, 0\right) \quad - \quad 1)$$

where **n(ℓ,ℓ')** is the frequency of concepts ℓ and ℓ' appear together, **n(ℓ)** is the frequency of concept ℓ and **N** is the total number of instances in the background data.

To determine the co-occurrence frequency, the number of unique combinations of objects in each image is counted across both datasets. These counts are then summed across all images before applying the PMI formula. The co-occurrence frequency for a pair of concepts is calculated by multiplying n(ℓ) and n(ℓ'). For self-occurrences (i.e., when a concept appears more than once in an image), the 'handshake' equation.

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

is used. To make it clearer, we provide one small example based on a sample image on Figure 6 and Table 1.

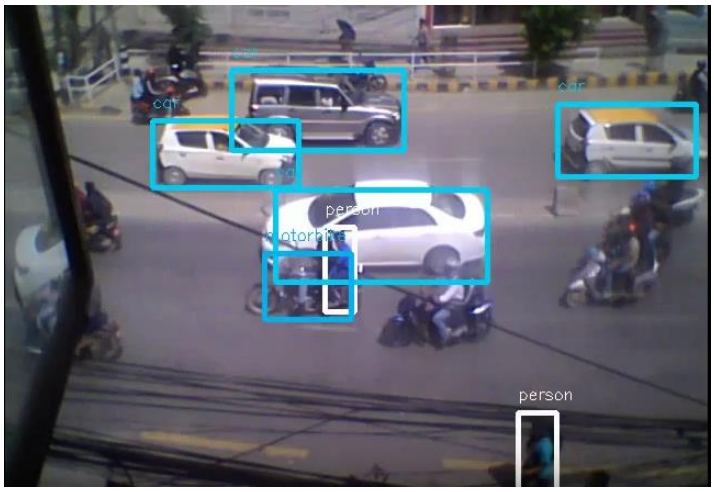

Figure 6: Annotated Voc image

n(car)	**4**
n(motorbike)	**1**
n(person)	**2**
n(car, car)	**6**
n(motorbike, motorbike)	**0**
n(person, person)	**1**
n(car, motorbike)	**4**
n(car, person)	**8**
n(motorbike, car)	**4**
n(motorbike, person)	**2**
n(person, car)	**8**
n(person, motorbike)	**2**

Table 1: Frequency of (co)-occurrences as counted from Figure 6

The authors use two different variants for this approach. First one is called **KF-All** which combines all the training sets to compute consistency matrix. On the other hand, second variant **KF-500** samples only images from the training set to show the impact of the amount of data on the performance of this approach. Therefore, we use both variants in our experiments as well.

Graph-based knowledge

The Knowledge Graph-based approach offers a more generalized method for determining semantic consistency between concepts. Unlike simple co-occurrence methods, a knowledge graph can capture complex relationships between concepts, even when they are not directly connected, enhancing the robustness of the analysis. To quantify these relationships as semantic consistency, the "random walk with restart" (RWR) algorithm [14] is used. This algorithm simulates a random walk from a starting node (or concept) through neighboring nodes in the knowledge graph. At each step, there's a probability of returning to the starting node, which helps prevent the walk from becoming trapped in localized areas of the graph. Over time, the probabilities of reaching different nodes stabilize, forming a steady-state distribution.

The RWR algorithm is mathematically represented as:

$$\vec{r} \leftarrow (1-c)\widetilde{W}\vec{r} + c\vec{e} \tag{3}$$

$\vec{r}$ **:** relatedness vector with respect to the starting node/seed $\vec{e}$

W : normalized weighted matrix that represents the weighted knowledge graph

c : restart probability constant.

In this work, an existing implementation of the RWR algorithm is utilized [15], along with the MIT ConceptNet version 5.7 [16] as the knowledge graph. The graph is filtered to include only the English segment, excluding negative relations (like "NotDesires," "NotHasProperty," etc.) and self-loops (where a concept is related to itself). The final graph consists of 1.16 million unique concepts and 3.35 million relations. For compatibility with the RWR algorithm, the graph's concept strings are converted to

integers using a Pandas DataFrame structure. It's important to note that ConceptNet's relations are directed, so if a relation is bi-directional, it's treated as two separate assertions. Each relation also comes with a weight that indicates its significance, which is considered in the RWR calculations. The relatedness vector is computed for all classes in the VOC (20 classes). This approach is called **KG-CNet57**.

4.2 Re-optimization

The re-optimization process involves fine-tuning detection results based on a cost function that takes into account semantic consistency and the original detection outputs. This process can be broken down into two main components as described by the cost function in equation 4 [1].

$$E(\widehat{P}) = (1-\epsilon)\sum_{b=1}^{B}\sum_{\substack{b'=1 \\ b'\neq b}}^{B}\sum_{\ell=1}^{L}\sum_{\ell'=1}^{L} S_{\ell,\ell'}\left(\widehat{P}_{b,\ell} - \widehat{P}_{b',\ell'}\right)^2 + \epsilon\sum_{b=1}^{B}\sum_{\ell=1}^{L} B\left\|S_{\ell,*}\right\|_1\left(\widehat{P}_{b,\ell} - P_{b,\ell}\right)^2 \quad (4)$$

It involves minimizing a cost function that has two main components:

1. **Semantic Consistency**: Ensures that the detection probabilities for different bounding boxes are consistent with the semantic relationships between the objects they represent. If two objects are semantically related, their detection probabilities should be similar.
2. **Original Prediction Alignment**: Ensures that the adjusted detection probabilities don't deviate too far from the original model predictions. This maintains the reliability of the initial model output. When ϵ = 1, we obtain the original predictions and decreasing ϵ increases the weight of the knowledge-aware predictions.

The process iteratively updates the detection probabilities using these components, usually converging within 30 iterations. To make this efficient, the method only considers the nearest bounding boxes and most relevant concepts for each object, leveraging sparse matrix operations to speed up calculations. This leads to object detection results that are more context-aware and aligned with real-world relationships between objects.

5. Implementation

5.1 Dataset

The method is tested on PASCAL VOC 2007 dataset [5]. The VOC dataset has 10k images in total and these are divided into 2.5k training set, 2.5k validation set and 5k test set. The data is annotated for 20 different classes. Although authors used both COCO and VOC dataset for the evaluation, because of the source and time-limit we decide to experiment on VOC only.

5.2 Metrics

For the evaluation, we use **MeanAveragePrecision** from **torchmetrics.detection.mean_ap** [17]. To be consistent with the authors, mAP@100 and recall@100 are selected as the main metrics. We also report recall@10 and by object areas as small, medium and large.

In **torchmetrics.detection.mean_ap** [17], recall is calculated by evaluating the ratio of true positive detections to the total number of ground truth objects, as the model's predictions are considered at varying confidence thresholds. The threshold of Intersection over Union (IoU) with the ground truth is selected as 0.5 for a detection to be considered as true positive.

The **torchmetrics.detection.mean_ap** [17] metric in *torchmetrics* uses a comprehensive set of confidence thresholds to compute the precision-recall curve. Specifically, it evaluates the metric at 101 discrete confidence thresholds from 0.0 to 1.0, inclusive with a step size of 0.01. This approach aligns with the evaluation practices used in the COCO dataset, which is a standard in object detection benchmarking. The COCO evaluation protocol also uses multiple thresholds to compute mAP.

5.3 Model training

Beyond the original baseline model Faster R-CNN with VGG-16 backbone from torchvision library, we also employ more recent models like YOLOv8 from ultralytics implementation [20] and DETR from [19].

While we use pre-trained weights for FRCNN and DETR models, we fine-tune YOLOv8 on VOC dataset. Training details of YOLOv8 is below (Please find Appendix A for the training loss graphs):

Image size	640
Batch size	16
Number of epochs	5
Learning rate	0.01
Optimizer	SGD
Momentum	0.937
Weight decay	0.0005

Table 2: Training setting of YOLOv8

For FRCNN, we use the pre-trained weights provided by [18] since our sources was not enough to train such model with 2.5k images and for DETR we use the pre-trained weights of COCO available on [19] with label mapping to be able to use properly on VOC dataset. The baseline results for all 3 models on VOC dataset are below:

	mAP@100	**recall@100**
FRCNN	55.17	44.11
YOLOv8	74.08	70.43
DETR	56.35	47.50

Table 3: mAP@100 and recall@100 of all standard models

For the knowledge-aware modification, we keep the parameters suggested by authors with top 500 detections whose scores bigger than 1e-5. We choose three different ϵ which defines the trade-off between standard predictions and knowledge aware predictions in equation 4 as 0.7, 0.8 and 0.9 and experiment for all models using different combinations of ϵ and semantic consistency matrices that are knowledge-graph based (KG-CNet57) and frequency-based (KF-500, KF-All). The updates of equation 4

are performed for 10 iterations since the authors suggests that it already shows convergence. Bk and Lk are both set to 5.

6. Results

6.1 FRCNN

eps = 0.7	mAP @100	Recall @100	Recall @10	Recall@100 by area small	medium	large
FRCNN	55.17	44.11	42.26	17.41	35.70	50.84
KF-500	54.43	47.55	43.29	18.64	38.50	54.76
KF-All	54.58	47.83	**43.48**	**19.07**	38.82	55.04
KG-CNet57	52.80	**48.58**	43.00	17.27	**39.04**	**56.23**

Table 4: Comparison of our knowledge-aware variants with the baseline method FRCNN with ϵ = 0.7

eps = 0.8	mAP @100	Recall @100	Recall @10	Recall@100 by area small	medium	large
FRCNN	55.17	44.11	42.26	17.41	35.70	50.84
KF-500	54.71	47.32	43.36	18.33	38.31	54.50
KF-All	54.78	47.56	**43.47**	**18.83**	38.60	54.76
KG-CNet57	53.71	**48.50**	43.27	17.63	**39.07**	**56.06**

Table 5: Comparison of our knowledge-aware variants with the baseline method FRCNN with ϵ = 0.8

eps = 0.9	mAP @100	Recall @100	Recall @10	Recall@100 by area small	medium	large
FRCNN	55.17	44.11	42.26	17.41	35.70	50.84
KF-500	55.01	46.84	43.29	18.39	37.93	53.93
KF-All	55.02	46.93	43.33	**18.52**	38.01	54.05
KG-CNet57	54.45	**48.11**	**43.54**	17.84	**38.87**	**55.52**

Table 6: Comparison of our knowledge-aware variants with the baseline method FRCNN with ϵ = 0.9

The results in table 4, 5 and 6 partly supports the claim in [1]. Across all three epsilon values (0.9, 0.8, 0.7) with different knowledge-aware variants, the mAP@100 of the original FRCNN model is the best with a value of 55.17. The KF-All and KF-500 methods show a slight decrease in mAP compared to FRCNN, with the decline being more pronounced at lower epsilon values (0.8, 0.7) as there is more emphasis on the knowledge-aware predictions in these values.

KG-CNet57 shows the most significant drop in mAP, especially at lower epsilon values. This suggests that all knowledge aware methods, while enhancing recall, may introduce noise or overfit in a way that negatively impacts precision.

KF-500 and KF-All show improved recall compared to the baseline FRCNN, with KF-All performing slightly better than KF-500. This improvement is consistent across all epsilon values, indicating that the frequency-based approaches effectively capture and utilize the semantic relationships between classes.

KG-CNet57 consistently provides the highest Recall @100, particularly at lower epsilon values. This demonstrates that the knowledge graph-based method significantly enhances the model's ability to recall objects, possibly by leveraging more generalized semantic relationships that aren't captured by co-occurrence frequency alone.

KF-All and KF-500 are relatively stable improvements over the baseline FRCNN, particularly for recall metrics, though they slightly compromise on mAP.

KG-CNet57 significantly boosts recall, particularly for larger objects, but at the cost of reduced precision (as seen by the drop in mAP).

Small Objects seem to be challenging for the KG-CNet57 method, likely due to the nature of the knowledge graph relationships, which may not prioritize or enhance detection for small-scale features.

eps = 0.9	mAP	Recall@100 by concepts																				
	@100	all	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
FRCNN	55.17	44.11	44.05	47.56	37.85	32.90	31.81	50.16	51.55	50.54	31.45	48.45	40.70	48.26	50.33	49.49	45.82	31.13	43.67	47.98	49.44	49.11
KF-500	55.01	46.84	**51.64**	51.18	43.25	32.62	32.89	51.89	54.63	56.84	31.32	51.70	40.64	53.56	**55.11**	52.55	45.38	33.50	47.49	47.37	54.34	48.81
KF-All	55.02	46.93	50.26	50.80	42.78	36.97	33.03	52.28	54.22	55.59	31.39	52.22	**41.04**	53.57	53.47	52.87	45.49	33.40	47.04	47.50	54.30	50.44
KG-CNet57	54.45	**48.11**	50.93	**51.93**	**43.30**	**38.02**	**33.06**	**52.87**	**55.91**	**57.57**	**34.15**	**53.28**	40.57	**54.57**	53.16	**53.82**	**49.15**	**34.22**	**47.68**	**50.51**	**56.42**	**51.16**

Table 7: Comparison of our knowledge-aware variants for all categories with the baseline method FRCNN with $\epsilon = 0.9$

Next, we present the comparison of results for all VOC categories for $\epsilon = 0.9$ as it can be seen in table 7. The reason for this choice is to be consistent with the authors (see Appendix B). Both KF-All and KG-CNet beat FRCNN in recall@100 by 2.82 and 4 points, respectively, with slightly affecting mAP. In particular, KF-All outperforms the baseline in 17 out of the 20 concepts, whereas KG-CNet outperforms the baseline in 19 concepts which are pretty consistent with the original results presented in Appendix B.

6.2 YOLOv8

eps = 0.7	mAP	Recall		Recall@100 by area		
	@100	@100	@10	small	medium	large
YOLOv8	74.08	**70.43**	**65.22**	**31.30**	**61.21**	**78.47**
KF-500	73.48	69.08	64.48	28.64	59.62	77.40
KF-All	73.65	69.27	64.68	28.88	59.61	77.61
KG-CNet57	72.38	68.41	63.49	27.04	58.60	76.85

Table 8: Comparison of our knowledge-aware variants with the baseline method YOLOv8 with $\epsilon = 0.7$

eps = 0.8	mAP @100	Recall @100	@10	Recall@100 by area small	medium	large
YOLOv8	74.08	**70.43**	**65.22**	**31.30**	**61.21**	**78.47**
KF-500	73.69	69.25	64.69	28.93	59.78	77.49
KF-All	73.82	69.44	64.83	29.16	59.97	77.68
KG-CNet57	73.01	68.83	63.97	27.62	59.15	77.22

Table 9: Comparison of our knowledge-aware variants with the baseline method YOLOv8 with ϵ = 0.8

eps = 0.9	mAP @100	Recall @100	@10	Recall@100 by area small	medium	large
YOLOv8	74.08	**70.43**	**65.22**	**31.30**	**61.21**	**78.47**
KF-500	73.89	69.36	64.81	29.19	59.91	77.60
KF-All	73.94	69.51	64.93	29.11	60.00	77.78
KG-CNet57	73.57	69.12	64.49	28.47	59.39	77.49

Table 10: Comparison of our knowledge-aware variants with the baseline method YOLOv8 with ϵ = 0.9

Unlike FRCNN, YOLOv8 consistently shows a slightly higher performance in all metrics across all ϵ values compared to the other methods, indicating that suggesting knowledge-aware methods does not show any improvement for YOLOv8.

KF-500 and KF-All are close in performance but consistently trail behind YOLOv8, showing marginal decreases in mAP. KG-CNet57 has the lowest mAP among the methods, particularly when ϵ is lower (72.38 at ϵ =0.7), suggesting that the more YOLOv8 utilize from knowledge graph-based approach, the lower performance it achieves.

Same observations are valid for both Recall@100 and Recall@10 across all ϵ values as well where YOLOv8 has 70.43 Recall@100 and 65.22 Recall@10, which are the highest among all approaches.

6.3 DETR

eps = 0.7	mAP @100	Recall @100	@10	Recall@100 by area small	medium	large
DETR	56.35	**47.50**	**46.60**	**22.66**	**37.24**	**53.02**
KF-500	34.69	28.61	28.48	14.68	23.48	31.16
KF-All	34.69	28.61	28.48	14.68	23.48	31.16
KG-CNet57	33.65	28.63	28.50	14.48	23.50	31.18

Table 11: Comparison of our knowledge-aware variants with the baseline method DETR with ϵ = 0.7

eps = 0.8	mAP @100	Recall @100	@10	Recall@100 by area small	medium	large
DETR	56.35	**47.50**	**46.60**	**22.66**	**37.24**	**53.02**
KF-500	35.01	28.61	28.49	14.69	23.50	31.15
KF-All	35.01	28.61	28.49	14.69	23.50	31.15
KG-CNet57	34.36	28.62	28.49	14.50	23.47	31.16

Table 12: Comparison of our knowledge-aware variants with the baseline method DETR with ϵ = 0.8

eps = 0.9	mAP @100	Recall @100	@10	Recall@100 by area small	medium	large
DETR	56.35	**47.50**	**46.60**	**22.66**	**37.24**	**53.02**
KF-500	35.45	28.66	28.53	14.71	23.59	31.14
KF-All	35.45	28.69	28.53	14.71	23.59	31.14
KG-CNet57	35.08	28.69	28.53	14.67	23.57	31.16

Table 11: Comparison of our knowledge-aware variants with the baseline method DETR with ϵ = 0.9

DETR also show similar trend as YOLOv8. It outperforms the other methods in all metrics across all ϵ values, with mAP@100 at 56.35, recall@100 at 47.50 and recall@10 at 46.60. KF-500, KF-All, and KG-CNet57 show similar recall and mAP values, all significantly lower than DETR's, suggesting that these methods make the performance of DETR much worse.

7. Conclusion

Since the source code of the original paper were not published, some missing details made it difficult for a fair comparison and re-implementation. The final value for parameter ϵ was not mentioned in the original paper (The authors of the replicated paper [18] confirmed the value of 0.9 by reaching them privately). Our limited sources were another drawback to produces the semantic consistency matrices since only knowledge-graph based requires 70 hours of running.
Our results based on some assumptions only partly supports the claim of the authors. Based on the results presented in tables 4, 5, 6 for FRCNN, KG-CNet57 is the best approach to maximize the recall. However, in all three approaches, there is a decrease in mAP suggesting that achieving improved recall without sacrificing from precision is not possible and the claim of the authors cannot be confirmed in our implementation.

Beyond the scope of this paper, we also experiment on more recent object detection models considering that the paper was published in 2017. We chose two models: YOLOv8 and DETR and tried both knowledge aware variants for three different ϵ. However, in both models, we couldn't get any performance improvement. Moreover, both methods caused severe performance drop compared to the standard detection results of YOLOv8 and DETR suggesting that knowledge-aware optimization is specific to each object detection model and the results on FRCNN cannot be generalized for other recent models.

8. 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. 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.

3. R. Varghese and S. M., "YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness," 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS), Chennai, India, 2024, pp. 1-6, doi: 10.1109/ADICS58448.2024.10533619.

4. Carion N., Massa F., Synnaeve G., Usunier N., Kirillov A., Zagoruyko S., “End-to-end object detection with transformers.” In: ECCV, Springer, 2020, pp. 213–229.

5. 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.

6. E. Krupka and N. Tishby. “Incorporating prior knowledge on features into learning.” In: Journal of Machine
Learning Research. Vol. 2. 2007, pp. 227–234.

7. Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. Large-scale object classification using label relation graphs. In ECCV, Part I, pages
48–64, 2014.

8. Carl Vondrick, Deniz Oktay, Hamed Pirsiavash, and Antonio Torralba. Predicting motivations of actions by leveraging text. In CVPR, pages 2997–3005, 2016.

9. Qi Wu, Peng Wang, Chunhua Shen, Anthony R. Dick, and Anton van den Hengel. Ask me anything: Free-form visual question answering based on knowledge from external sources. In CVPR, pages 4622–4630, 2016.

10. Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge J. Belongie. “Objects in context”. In ICCV, pp. 1–8, 2007.

11. Jongkwang Hong, Yongwon Hong, Youngjung Uh, and Hyeran Byun. Discovering overlooked objects: Context-based boosting of object detection in indoor scenes. Pattern Recogn. Lett., 86:56–61, 2017.

12. Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: a Web-scale approach to probabilistic knowledge fusion. In KDD, pages 601–610, 2014.

13. G. Bouma. “Normalized (Pointwise) Mutual Information in Collocation Extraction.” In: Proceedings of German Society for Computational Linguistics (GSCL 2009) (2009), pp. 31–40.
14. H. Tong, C. Faloutsos, and J.-Y. Pan. Fast Random Walk with Restart and Its Applications. Tech. rep.

15. J. Jung. Python Implementation for Random Walk with Restart (RWR). 2021.

16. Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. "ConceptNet 5.5: An Open Multilingual Graph of General Knowledge." In proceedings of AAAI 31.

17. Falcon, W., & The PyTorch Lightning team. (2019). PyTorch Lightning (Version 1.4) [Computer software]. https://doi.org/10.5281/zenodo.3828935

18. Lemmens J., Jancura P., Dubbelman G., Elforai H., “[Re] Object Detection Meets Knowledge Graphs” ReScience C 4.1, 2018.

19. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. Retrieved from https://github.com/facebookresearch/detr

20. Jocher, G., Chaurasia, A., Millard, A., et al. (2023). Ultralytics YOLOv8: Cutting-Edge, Real-Time Object Detection. Retrieved from https://github.com/ultralytics/ultralytics

APPENDICES

A Training Loss

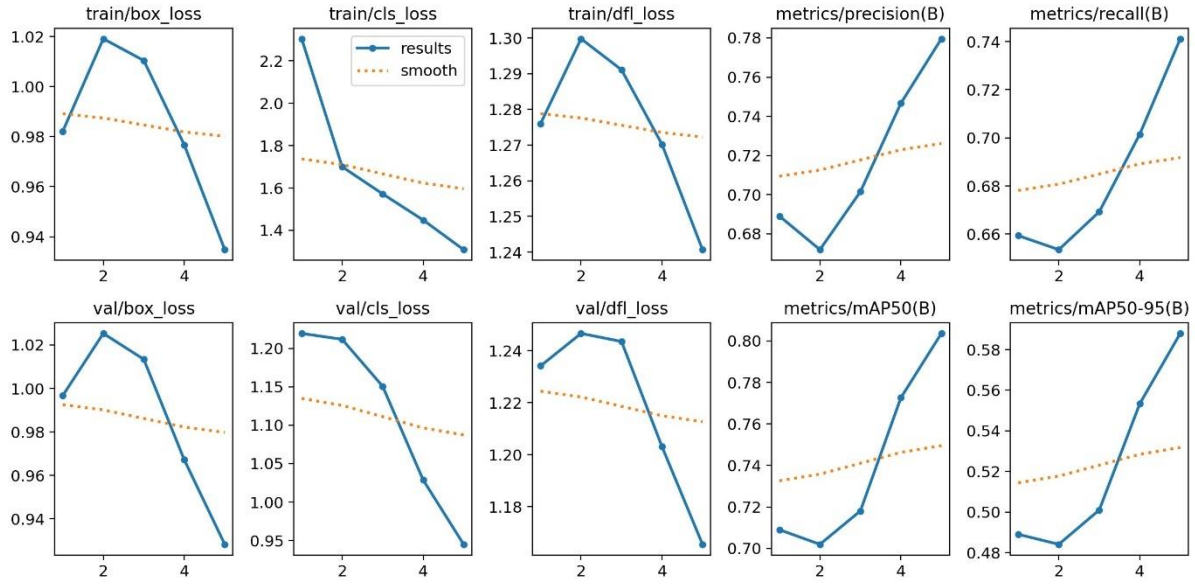

Figure 1: Training and validation box, dfl and cls losses of YOLOv8 for training

B Original Results

	mAP @100	Recall@100 by concepts																			
		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**

Table 1: Original Comparisons for FRCNN presented by [1]