# Detecting 11K Classes: Large Scale Object Detection without Fine-Grained Bounding Boxes - Supplemental File

Hao Yang Hao Wu Hao Chen
Amazon Web Services
{haoyng, goodwu, hxen}@amazon.com

## 1. Details of the Memory Module

To represent the evolving feature space and the current overall model inference uncertainty, memory update is performed every iteration to accommodate the most recent updates of the network. To ensure the quality of memories, we only utilize labelled data to update the memory. We have two types of updates for our Dual-Level Memory (DLM) module: coarse-grained proposal level memory update and fine-grained image level memory update. For the former update, we use proposal-level scores and features to update the corresponding memory slots, and for the latter update, we use image-level scores and features.

Specially, assume there are $n_j$ proposals/images from the $j$-th class with their feature vectors and probabilistic predictions as $\{(\boldsymbol{x}_i, \boldsymbol{p}_i)\}_i^{n_j}$, the $j$-th memory slot $(\boldsymbol{k}_j, \boldsymbol{v}_j)$ is cumulatively updated over all the training iterations as follows:

$$\begin{cases} \boldsymbol{k}_j \leftarrow \boldsymbol{k}_j - \eta \nabla \boldsymbol{k}_j \\ \boldsymbol{v}_j \leftarrow \frac{\boldsymbol{v}_j - \eta \nabla \boldsymbol{v}_j}{\sum_{i=1}^{C} \boldsymbol{v}_{j,i} - \eta \nabla \boldsymbol{v}_{j,i}} \end{cases}, \text{with} \tag{1}$$

$$\begin{cases} \nabla \boldsymbol{k}_j = \frac{\sum_{i=1}^{n_j} (\boldsymbol{k}_j - \boldsymbol{x}_i)}{1+n_j} \\ \nabla \boldsymbol{v}_j = \frac{\sum_{i=1}^{n_j} (\boldsymbol{v}_j - \boldsymbol{p}_i)}{1+n_j} \end{cases} \tag{2}$$

where $\boldsymbol{k}_j$ is the *key embedding*, i.e., the feature representation of the class $j$ in the feature space, and $\boldsymbol{v}_j$ is the *value embedding*, i.e., the mutli-class probabilistic prediction w.r.t. class $j$. $\eta$ is the learning rate. The value $\boldsymbol{v}_j$ is normalized to ensure its probability distribution nature. The key and value embeddings are initialized with $\mathbf{0}$ and a uniform vector, respectively.

### 1.1. Assimilation and Accommodation

We use an assimilation-accommodation mechanism [2] for our semi-supervised detection. *Memory Assimilation* computes the memory prediction for each training sample by key addressing and value reading. This process can be interpreted as cluster assignments. *Memory Accommodation* computes the memory loss to formulate the final semi-supervised learning objectives. We introduce the details as follows.

**Assimilation:** For a proposal or image with the feature represented as $\boldsymbol{x}$, and with label $j$, the memory assimilation process is essentially computing the memory prediction $\hat{\boldsymbol{p}}$ by the weighted sum of all value embeddings as follows:

$$\hat{\boldsymbol{p}} = \sum_{i=1}^{C} w(\boldsymbol{k}_i|\boldsymbol{x}) \boldsymbol{v}_i, \tag{3}$$

where $w(\cdot)$ is an assignment function. We can use hard assignment if the image/proposal label is known, i.e.,

$$w(\boldsymbol{k}_i|\boldsymbol{x}) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Or we use distance-based soft assignment for unlabelled images/proposals, i.e.,

$$w(\boldsymbol{k}_i|\boldsymbol{x}) = \frac{e^{-\beta \|\boldsymbol{x} - \boldsymbol{k}_i\|_2}}{\sum_j e^{-\beta \|\boldsymbol{x} - \boldsymbol{k}_j\|_2}}. \tag{5}$$

**Accommodation:** Given the network prediction $\boldsymbol{p}$ and the memory prediction $\hat{\boldsymbol{p}}$ for a proposal/image, the memory loss is defined as:

$$L_m = H(\hat{\boldsymbol{p}}) + D_{KL}(\boldsymbol{p}||\hat{\boldsymbol{p}}), \tag{6}$$

where $H(\cdot)$ is the entropy and $D_{KL}(\cdot)$ is the Kullback-Leibler (KL) divergence.

## 2. Design Choices on Pooling Strategies

| mAP$^{50}$ | Feature Maps | Score Maps |
|---|---|---|
| Coarse-grained | **52.9** | 52.2 |
| Fine-grained | **49.2** | 47.8 |

Table 1: Shall we pool on feature or score maps?

In this section, we study different options in the network design. For example, while doing the RoI pooling, should

we pool on feature maps or score maps? Which global pooling function should we use for aggregating proposal scores to image level scores? We run experiments on the OpenImages dataset to search for the best options for these questions. Note that these experiments are without using the memory module, thus their results are lower than our final results.

First, although we can follow the literature and pool feature maps for the RoI pooling, we also run experiments on score maps. We see from Table 1 that pooling on feature maps provides slightly better results than on score maps.

Second, we study different types of global pooling methods for aggregating the proposal scores to image level scores in weakly supervised stream. Max, average, sum and top-k pooling are the options that we investigate. Unfortunately, we cannot get meaningful results with average and sum pooling. These pooling methods basically treat each proposal equally thus are hard to tune. We also study the WSDDN [1] type of pooling (essentially a self-weighted sum pooling), but cannot get meaningful results neither. We find out that for large scale weakly-supervised problem, using max and top-k pooling are much easier to train. Results are demonstrated in Table 2, Top-5 pooling achieves slightly better performance than the max pooling.

| $mAP^{50}$ | Max | Top-5 Average |
|---|---|---|
| Coarse-grained | 50.8 | **52.2** |
| Fine-grained | 46.1 | **47.8** |

Table 2: Global pooling methods for aggregating proposal scores in weakly supervised stream.

## 3. Additional Experiments on ImageNet and VOC-COCO

Though our framework is not designed for these settings, we show here our method can still outperform [4, 5] and DOCK [3]. Following [4, 5], we use the first 100 classes (in alphabetical order) in ImageNet detection set as the fully-supervised set and the remaining 100 classes as the weakly-supervised set. Following DOCK, we use the 20 classes from VOC as the fully-supervised set, and the remaining 60 (80-20) classes from COCO as the the weakly-supervised set. Results are summarized in Table 3 and 4.

## 4. Additional Qualitative Results

We show more qualitative results in Fig.1.

## References


[1] Hakan Bilen and Andrea Vedaldi. Weakly supervised deep detection networks. In *CVPR*, pages 2846–2854, 2016. 2


| Method | mAP-Fully | mAP-Weakly |
|---|---|---|
| [31] | 28.0 | 20.0 |
| [34] | - | 23.3 |
| Ours | **53.3** | **37.5** |

Table 3: Results on ImageNet Detection dataset where we evaluate fully-supervised and weakly-supervised classes separately. Note that we did not augment the training set by using higher quality validation data as [4, 5] did.

| Method | mAP | $mAP^S$ | $mAP^M$ | $mAP^L$ |
|---|---|---|---|---|
| DOCK | 14.4 | 2.0 | 12.8 | 24.9 |
| Ours | **18.8** | **5.3** | **13.8** | **31.2** |

Table 4: Results on VOC-COCO dataset where we evaluate weakly-supervised classes on overall mAP, as well as mAP for small, medium and large sizes objects.


[2] Yanbei Chen, Xiatian Zhu, and Shaogang Gong. Semi-supervised deep learning with memory. In *ECCV*, 2018. 1

[3] Krishna Kumar Singh, Santosh Kumar Divvala, Ali Farhadi, and Yong Jae Lee. DOCK: detecting objects by transferring common-sense knowledge. In *ECCV*, pages 506–522, 2018. 2

[4] Yuxing Tang, Josiah Wang, Boyang Gao, Emmanuel Dellandréa, Robert J. Gaizauskas, and Liming Chen. Large scale semi-supervised object detection using visual and semantic knowledge transfer. In *CVPR*, pages 2119–2128, 2016. 2

[5] Jasper R. R. Uijlings, Stefan Popov, and Vittorio Ferrari. Revisiting knowledge transfer for training object class detectors. In *CVPR*, pages 1101–1110, 2018. 2

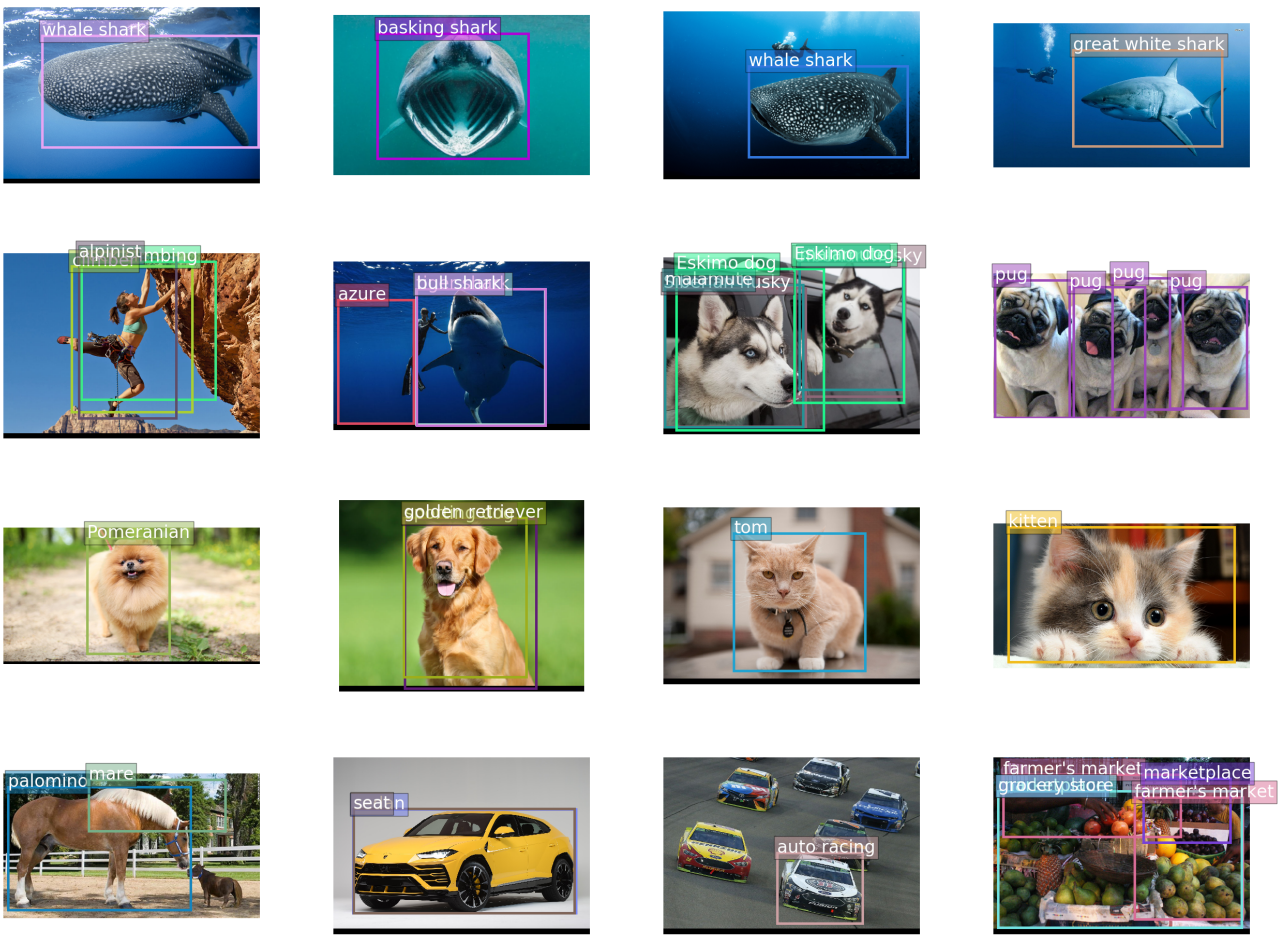


Figure 1: Qualitative Results on ImageNet-11K. Common failure cases include: 1) False classification; 2) Miss detection; 3) Confusing scenes and objects; and 4) Confusing parts with objects.