Deep Structured Instance Graph for Distilling Object Detectors
— *Supplementary Material* —

Yixin Chen[1], Pengguang Chen[1], Shu Liu[2], Liwei Wang[1], Jiaya Jia[1,2]
The Chinese University of Hong Kong[1] SmartMore[2]

Here we provide some additional details about what was left in the main text. First we show the algorithm of our method involved in the whole distillation procedure. Next, we elaborate the evaluation metrics of the COCO [2] benchmark and the student training configurations and settings. We also give definition of Kullback-Leibler Divergence Loss that was not mentioned in the submitted paper. At last, visualization results are provided to afford more intuitive understanding.

1. Algorithm

We show the pseudo algorithm of our method involved in detection models, including structured graph building and graph distillation loss calculation:

Algorithm 1 Graph Distillation Loss

Input: image x, target t, teacher T, student S
Output: Graph Distillation Loss

$\mathcal{F}_T \Leftarrow$ T.backbone(x)
$\mathcal{F}_S \Leftarrow$ S.backbone(x)
proposals $\Leftarrow$ S.rpn(x, t, $\mathcal{F}_S$)
Node$_T^{fg}$, Node$_T^{bg}$ $\Leftarrow$ RoIPool($\mathcal{F}_T$, proposals)
Node$_S^{fg}$, Node$_S^{bg}$ $\Leftarrow$ RoIPool($\mathcal{F}_S$, proposals)
Node$_T^{mine}$, Node$_S^{mine}$ $\Leftarrow$ BackGroundMining(Node$_T^{bg}$, Node$_S^{bg}$, T)
Node$_T$ $\Leftarrow$ cat(Node$_T^{fg}$, Node$_T^{mine}$)
Node$_S$ $\Leftarrow$ cat(Node$_S^{fg}$, Node$_S^{mine}$)
Edge$_T$ $\Leftarrow$ similarity_function(Node$_T$)
Edge$_S$ $\Leftarrow$ similarity_function(Node$_S$)
$L_{Node} \Leftarrow$ mseloss(Node$_T$, Node$_S$)
$L_{Edge} \Leftarrow$ mseloss(Edge$_T$, Edge$_S$)
return $L_{Node} + L_{Edge}$

where the algorithm of function BackGroundMining is detailed as follows:

Algorithm 2 BackGroundMining

Input: Node$_T^{bg}$, Node$_S^{bg}$, Teacher T
Output: Mined Background Nodes Node$_T^{mine}$, Node$_S^{mine}$

Initialize thrs = t
RoILoss$_T^{bg}$ $\Leftarrow$ T.RoIclsLoss(Node$_T^{bg}$)
NodeIndex $\Leftarrow$ SelectAboveThrs(RoILoss$_T^{bg}$, thrs)
Node$_T^{mine}$ $\Leftarrow$ SelectIndex(Node$_T^{bg}$, NodeIndex)
Node$_S^{mine}$ $\Leftarrow$ SelectIndex(Node$_S^{bg}$, NodeIndex)
return Node$_T^{mine}$, Node$_S^{mine}$

2. Experimental Details

We skip some details in the experiment of the main paper due to pages limitation. Here we present a brief description about the COCO evaluation metrics and the training settings.

2.1. Evaluation Metrics

Here we adopt the evaluation metric Average Precision(AP) in our experiments. AP is a quite popular evaluation metric in measuring object detector's performance, here we only refer to the COCO style AP. Before that, we first introduce IoU(Intersection over union) to determine whether one prediction is correct w.r.t. a ground truth object or not. Specifically, IoU defines the intersection between the predicted bounding box and ground truth bounding box:

$$IOU = \frac{area\ of\ overlap}{area\ of\ union} \tag{1}$$

We also set a IoU $threshold$ to classify whether a prediction is true positive or false positive. Precision actually means the percentage of true positive predictions in all predictions while recall is the percentage of true positive predictions in all positive samples:

$$\begin{aligned} Precision &= \frac{TP}{TP+FP} \\ Recall &= \frac{TP}{TP+FN} \end{aligned} \tag{2}$$

where TP, FP, FN represents true positive, false positive, and false negative respectively.

We rank all predictions in a descending order according to their confidences. With every rank, we can calculate the

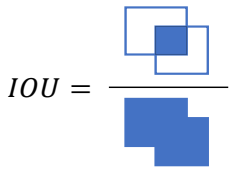


Figure 1. IoU

```
Average Precision (AP):
  AP               % AP at IoU=.50:.05:.95 (primary challenge metric)
  AP^IoU=.50       % AP at IoU=.50 (PASCAL VOC metric)
  AP^IoU=.75       % AP at IoU=.75 (strict metric)
AP Across Scales:
  AP^small         % AP for small objects: area < 32^2
  AP^medium        % AP for medium objects: 32^2 < area < 96^2
  AP^large         % AP for large objects: area > 96^2
```

Figure 2. AP series. https://cocodataset.org/#detection-eval.

precision and recall, which will finally form a Precision-Recall curve. In the COCO style, we sample 101 points recall@[0:0.01:1] and average all these corresponding precision to calculate mean AP. With different IoU $threshold$, true positive predictions can be different. In our experiments, we adopt six different kinds of AP according to various IoU $thresholds$ and object scales, descriptions are detailed in Figure 2.

2.2. Training Settings

For training, all our experiments are performed on 4 Nvidia RTX 2080Ti GPUs, and all the student models are sticking to the 1x/3x(around 12/37 epochs) COCO training schedule. Specifically, we set batch size as 8 with 2 images allocated to each GPU and double the training iterations to 180000/540000 for 1x/3x training. Our student models are optimized with Stochastic Gradient Descent(SGD) with the base learning rate 0.01 and we reduce it by gamma 0.1 at iteration (120000, 160000)/(420000, 500000) for 1x/3x training. All our backbone models are initialized with weights pre-trained on ImageNet.

3. Kullback-Leibler(KL) Divergence Loss

The definition of KLD loss is as follows:

$$\begin{aligned} D_{KL}(p_t|p_s) &= -\sum_i p(x_{t,i}) \log p(x_{s,i}) \\ &= \sum_{x_t, x_s} p(x_{t,i}) \log \frac{p(x_{t,i})}{p(x_{s,i})} \end{aligned} \tag{3}$$

where x_t, x_s represents the logits of teacher and student respectively, and probability

$$p(x_{\cdot,i}) = \frac{\exp(x_{\cdot,i})}{\sum_j \exp(x_{\cdot,j})} \tag{4}$$

Actually, in our experiment, following [1], we use the logits softened by temparature T, so the probability is:

$$\hat{p}(x_{\cdot,i}) = \frac{\exp(x_{\cdot,i}/T)}{\sum_j \exp(x_{\cdot,j}/T)} \tag{5}$$

since the gradients produced by soft logits are also scaled as $1/T^2$, we should multiply them by T^2, so the KLD loss is transformed to:

$$\begin{aligned} D_{KL}(p_t|p_s) &= -\sum_i \hat{p}(x_{t,i}) \log \hat{p}(x_{s,i}) * T^2 \\ &= \sum_{x_t, x_s} \hat{p}(x_{t,i}) \log \frac{\hat{p}(x_{t,i})}{\hat{p}(x_{s,i})} * T^2 \end{aligned} \tag{6}$$

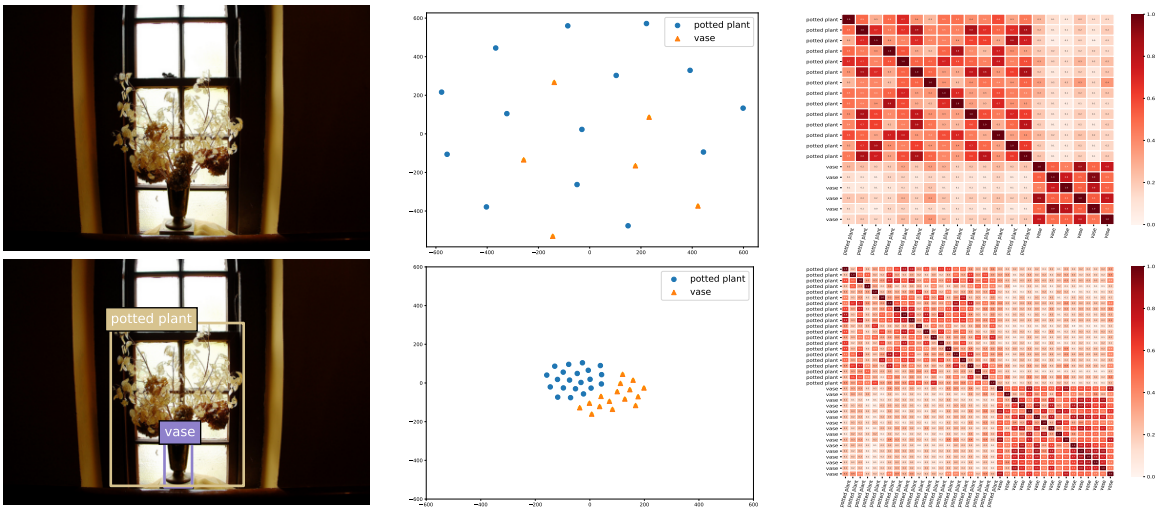
In our experiments, we simply set T to **1**.

4. Visualization Results

Graph Visualization: We show more visualization results on COCO test images. As is shown in Figure 3. It can be seen that from node visualization, the scattered nodes from several categories are mixing with each other, which means student's feature space is quite different from that of teacher. Also, in edge visualization, some edges in student still reveal strong intensities among nodes in different categories while edges in teacher almost show strong links among nodes in the same categories. These appearing visualization results enlighten us that there are great gaps between the knowledge space of student and teacher, which propels us to conduct graph transferring to close that.

Object Detection Visualization: To intuitively illustrate how our distillation method makes a difference to improve student detector's performance, we visualize student detection results before distillation and the counterparts distilled after. See Figure 4. Before distillation, student detectors with backbones in small capacity show poor results including missing boxes, duplicate boxes, and inaccurate boxes *etc*. However, refined with our method, students with substantial promotions are able to bridge the gaps and behave much more accurate in classification and localization.

Instance Segmentation Visualization: We also show the visualization of distillation performance in instance segmentation. As is shown in Figure 5, the students are not behaving well in classification and localization, and even show the poor segmentation results in each predicted mask. As for teachers, the segmentation results seem better, yet some wrongly predicted pixels still exist. Since our method has added more regularization than normal baseline models, the masks in our method have exhibited better shapes and finer borders than the student baseline, even beyond the teacher in some test cases.



(a) Labeled Image (b) Node Visualization (c) Edge Visualization

(d) Labeled Image (e) Node Visualization (f) Edge Visualization

(g) Labeled Image (h) Node Visualization (i) Edge Visualization

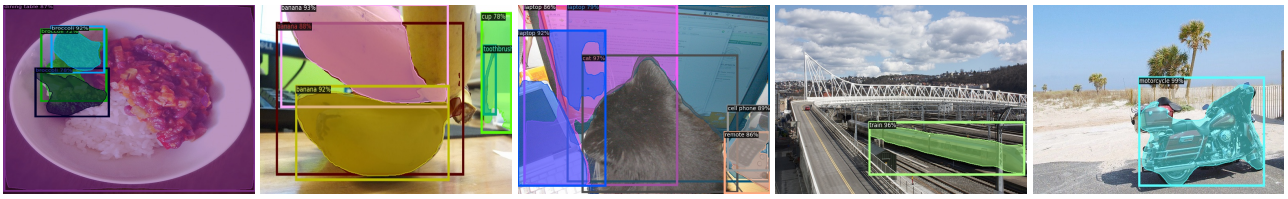
Figure 3. Graph visualization on test image.

(a) Student **Before** Distillation

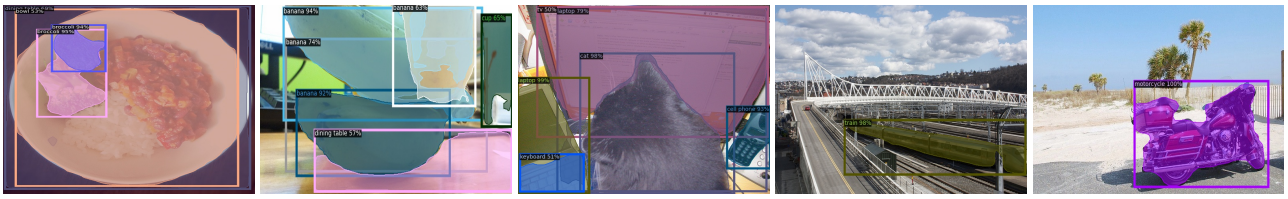
(b) Student **After** Distillation

(c) Teacher

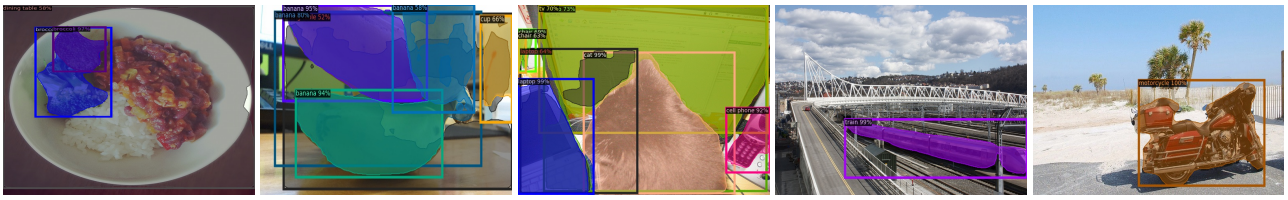
Figure 4. Object detection distillation visualization on COCO2017 `test`. We adopt (a) Student Faster R-CNN with ResNet18, (b) Student distilled using our method, and (c) Teacher ResNet50 as our trained models for visualization.



(a) Student **Before** Distillation



(b) Student **After** Distillation



(c) Teacher

Figure 5. Instance segmentation distillation visualization on COCO2017 `test`. We adopt (a) Student Faster R-CNN with ResNet18, (b) Student distilled using our method, and (c) Teacher ResNet50 as our trained models for visualization.

References

[1] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015. 2

[2] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014. 1