Semi-Supervised and Long-Tailed Object Detection with CascadeMatch

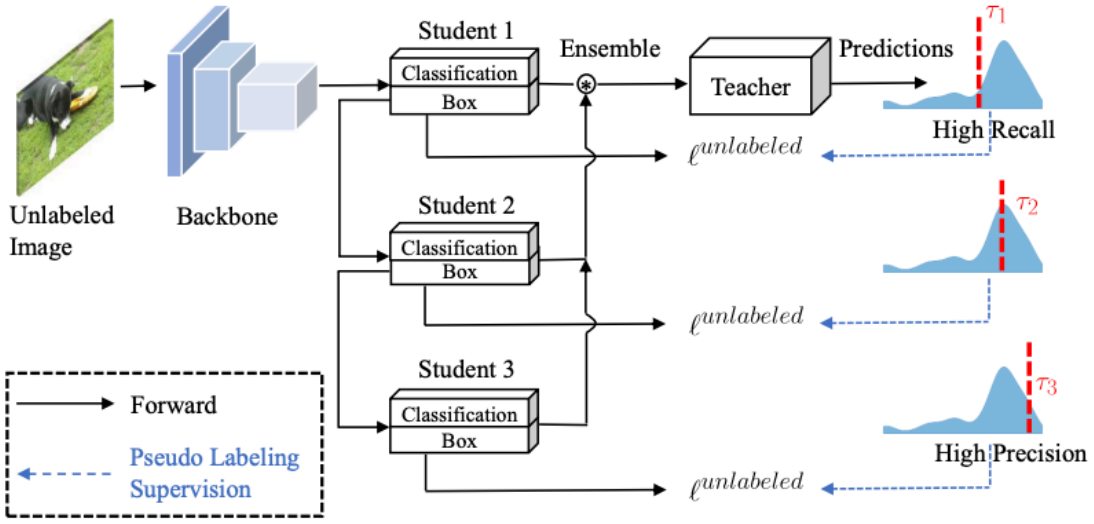
Background:

- Long tailed object detection:
  - Most SSOD studies are on coco dataset, which has highly balanced data distributions.
  - Real world data - Majority of the classes have only few labelled examples.
  - LVIS v1.0 dataset : Comprises of 3 class groups
    - Rare : if the classes have images between [1-10).
    - common : if the classes have images between [10-100).
    - Frequent : if the classes have images between [100-).
  - When a prominent SSOD **Unbiased Teacher** is trained on a long-tailed dataset LVIS v1.0, the authors of the paper noticed the following issues :
    - Using a fixed threshold fails to give a good trade-off between precision and recall
    - Frequent classes have higher confidence scores
    - Model's exposure to the rare and common classes is substantially reduced compared to the frequent classes when a fixed threshold is used.

Key Ideas:

Cascade Match

▼ Cascade Match



(a) Pipeline of CascadeMatch

▼ **Cascaded Pseudo labelling**

- The model has cascaded detection heads. The idea is to apply progressive thresholds for filtering the pseudo-labels for the detection heads.
- For a given image, the RPN generates the object proposal, send the proposal to the first head. It outputs a bounding box and class probability.
- The next head takes the box predicted by the previous head as input and outputs a refined box, and a new class probability.
- In the scenario when we have labels, the training is simple. Each head is supervised using the ground truth.
- In case of a unlabelled image :
    - The predictions (both confidence scores and bounding boxes) of the heads are aggregated by mean.

$$p_t = \frac{1}{K}\sum_{k=1}^{K} p_k(y|\boldsymbol{x}, \boldsymbol{b}_{k-1}) \quad \text{and} \quad \boldsymbol{b}_t = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{b}_k,$$

        - The notation above is for a single proposal.
    - p_t is a list of average confidence scores per class. The max and argmax of which gives us the pseudo confidence score and pseudo label.
    - These pseudo labels are used to the train the network.
    - The pseudo-labels are filtered using a threshold which is different for each head.

▼ Selecting the threshold (**Adaptive pseudo-label mining**):

- In labelled image scenario, the ensemble predictions made on each ground - truth class are looked at. The predictions are stratified with respect to class.
- Mean and standard deviation of the confidence scores per class is calculated. Then the threshold for each class per head is given by :

$$\tau_k^c = \mu_c + \sigma_c * \epsilon_k$$

$$\epsilon_k \in \{1, 1.5, 2\}$$

Threshold for each head is controlled by E_k .

Impressions

▼ Results and Ablation Studies

▼ **Cascaded Pseudo labelling Module** and **Adaptive pseudo-label mining**: on LVIS v1.0 validation set. unsupervised data : COCO unlabelled2017

CPL	APM	AP^{Fix}	AP_r^{Fix}	AP_c^{Fix}	AP_f^{Fix}
✗	✗	26.3	19.7	25.3	30.3
✓	✗	30.1	21.9	29.3	34.5
✗	✓	28.9	22.5	27.9	32.8
✓	✓	**30.5**	**23.1**	**29.7**	**34.7**

- The above table shows the model trained with and without the proposed CPL and APM modules. Top row is supervised.
- We can see that both CPL and APM individually contribute in improving the model performance.

▼ E_k hyperparameter

ϵ_1	ϵ_2	ϵ_3	AP^{Fix}	AP_r^{Fix}	AP_c^{Fix}	AP_f^{Fix}
0.0	0.0	0.0	29.8	21.7	29.1	34.1
0.0	1.0	2.0	30.2	**23.3**	29.2	34.3
1.0	2.0	3.0	30.3	22.6	29.5	34.4
1.0	1.5	2.0	**30.5**	23.1	**29.7**	**34.7**

▼ **K** : Number of cascade heads

K	$\mathrm{AP}^{\mathrm{Fix}}$	$\mathrm{AP}_{\mathrm{r}}^{\mathrm{Fix}}$	$\mathrm{AP}_{\mathrm{c}}^{\mathrm{Fix}}$	$\mathrm{AP}_{\mathrm{f}}^{\mathrm{Fix}}$	$\mathrm{T}_{\mathrm{train}}$
1	26.4	20.4	26.6	28.9	0.36
2	28.0	21.4	27.1	31.9	0.42
3	**30.5**	**23.1**	**29.7**	34.7	0.47
4	30.0	22.1	29.2	34.6	0.59
5	29.9	21.2	29.0	**34.9**	0.72

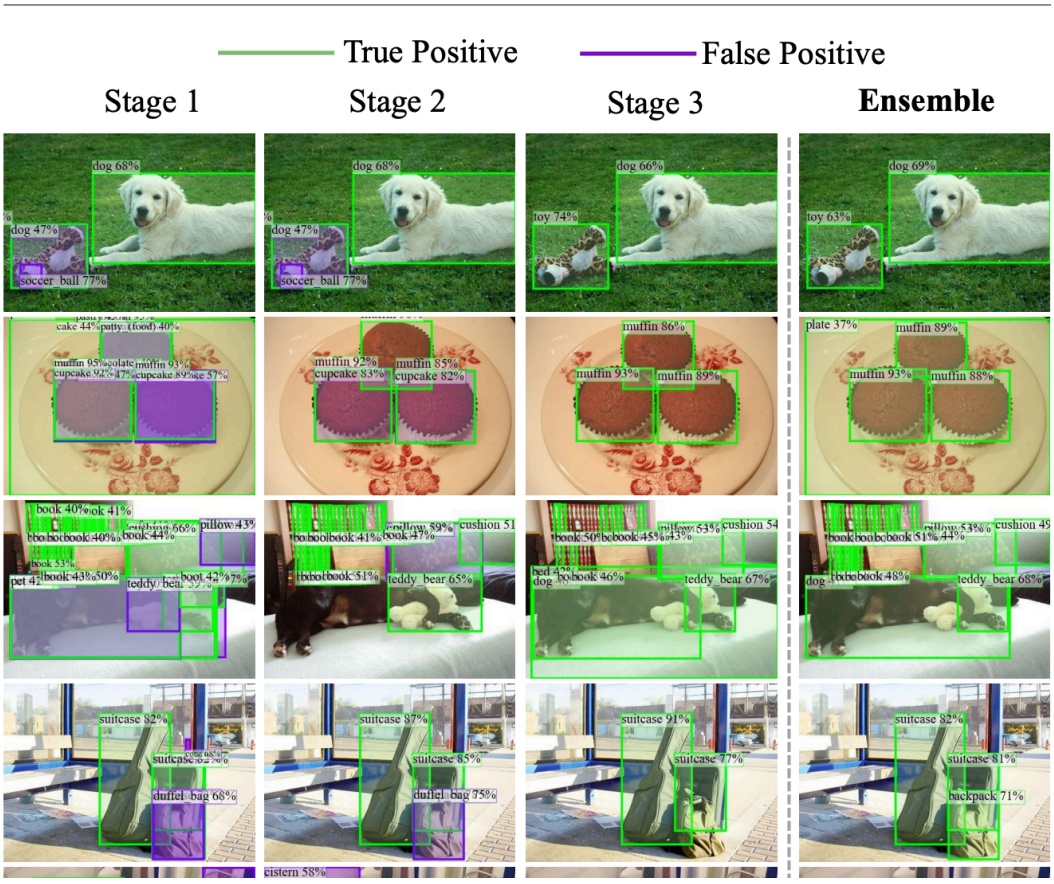
- The model performance keeps improving as the number of head increased from 1-3.
- The performance of rare and common classes is dropped if we continue to increase the k from 3 to 4 or 5, probably due to the over-fitting and undesired memorizing effects of few-shot classes as we increase the model capacity.

▼ Confirmation bias

- **Def** : Model iteratively reinforced by incorrect label produced by itself.
- To establish the cascaded pseudo labels are helping to reduce confirmation bias, they trained the network using the LVIS v1.0 dataset where they use 30% of the training set as labelled data and the rest 70% as the unlabelled data.
- From the table below we can see the accuracy of the individual heads and their ensemble at different stages of training. They observe that the pseudo label accuracy is consistently lower for individual heads when compared to the ensemble.

Iter.	60k	120k	180k
Head 0	32.8	51.5	67.3
Head 1	50.5	62.4	73.2
Head 2	55.1	71.0	84.1
Ensemble	**66.4**	**79.5**	**88.9**

▼ Qualitative



▼ Semi supervised model results comparison: On LVIS V1.0 Validation set.

Method	Framework	Backbone	Schedule	AP^{Fix}	AP_r^{Fix}	AP_c^{Fix}	AP_f^{Fix}
Supervised				26.3	19.7	25.3	30.3
CSD [26]				26.8	19.9	25.8	31.0
STAC [58]	Cascade R-CNN	R-50-FPN	12e	27.5	20.3	26.3	32.1
Unbiased Teacher [43]				28.6	20.8	27.9	32.8
Soft Teacher [80]				29.2	21.1	28.4	33.7
LabelMatch [9]				29.4	20.3	29.2	33.8
CascadeMatch (*ours*)				**30.5**	**23.1**	**29.7**	**34.7**
Supervised				27.1	20.3	26.1	31.1
Unbiased Teacher [43]	Cascade R-CNN	R-101-FPN	12e	31.0	24.6	30.2	35.0
CascadeMatch (*ours*)				**32.9**	**26.5**	**31.8**	**36.8**
Supervised				31.7	23.5	29.5	38.0
Unbiased Teacher [43]	Sparse R-CNN	PVT	30e	33.5	24.6	31.4	40.2
CascadeMatch (*ours*)				**35.2**	**27.5**	**33.2**	**41.1**

- When comparing the Cascade match with different SSOD networks, it consistently outperforms the other SSODs with non-trivial margins.
- * The SSODs use coco 2017 unlabelled set as the unlabelled dataset.
- on COCO-LT validation set:

Method	AP	AP_1	AP_2	AP_3	AP_4
Supervised	25.4	2.5	16.2	29.9	33.7
CSD	25.9 (+0.5)	2.0	15.2	32.1	34.0
STAC	26.4 (+1.0)	2.2	16.3	32.4	34.1
UT	26.7 (+1.3)	2.2	18.0	31.8	34.3
Ours	**27.8** **(+2.4)**	**4.0**	**20.4**	**32.4**	**34.5**