Supplementary Material

In this supplementary material, we provide additional details which we could not include in the main paper due to space constraints, including experimental analysis, implementation details, discussion and results that help us develop further insights to the proposed Open World Object Detection approach. We discuss:

- Sensitivity analysis on queue size of Feature Store, the momentum parameter η, margin in clustering loss Δ and temperature parameter in energy computation.
- Additional details on contrastive clustering
- More specific implementation details.
- Discussion regarding failure cases.
- Related works in incremental object detection.
- Some qualitative results of ORE.

A. Varying the Queue Size of $\mathcal{F}_{Store}$

In Sec. 4.1, we explain how class specific queues $\boldsymbol{q}_i$ are used to store the feature vectors, which are used to compute the class prototypes. A hyper-parameter Q controls the size of each $\boldsymbol{q}_i$. Here we vary Q, while learning Task 1, and report the results in Tab. 7. We observe relatively similar performance, across experiments with different Q values. This can be attributed to the fact that after a prototype is defined, it gets periodically updated with newly observed features, thus effectively evolving itself. Hence, the actual number of features used to compute those prototypes ($\mathcal{P}$ and $\mathcal{P}_{new}$) is not very significant. We use Q $= 20$ for all the experiments.

Q	WI (↓)	A-OSE (↓)	mAP (↑)
5	0.02402	8123	56.01
10	0.02523	8126	56.02
20	0.02193	8234	56.34
30	0.02688	8487	55.78
50	0.02623	8578	56.22

Table 7: We find that varying the number of features that are used to compute the class prototype does not have a huge impact on the performance.

B. Sensitivity Analysis on η

The momentum parameter η controls how rapidly the class prototypes are updated, as elaborated in Algorithm 1. Larger values of η imply smaller effect of the newly computed prototypes on the current class prototypes. We find from Tab. 8 that performance improves when prototypes are updated slowly (larger values of η). This result is intuitive, as slowly changing the cluster centers helps stabilize contrastive learning.

η	WI (↓)	A-OSE (↓)	mAP (↑)
0.4	0.05926	9476	55.96
0.6	0.04977	9095	55.56
0.8	0.02945	8375	55.73
0.9	0.02193	8234	56.34

Table 8: We see that higher values of η gives better performance, implying that gradual evolution of class prototypes improves contrastive clustering.

C. Varying the Margin (Δ) in $\mathcal{L}_{cont}$

The margin parameter Δ in the contrastive clustering loss $\mathcal{L}_{cont}$ (Eqn. 1) defines the minimum distance that an input feature vector should keep from dissimilar class prototypes in the latent space. As we see in Tab. 9, increasing the margin while learning the first task, increases the performance on the known classes and how unknown classes are handled. This would imply that larger separation in the latent space is beneficial for ORE.

Δ	WI (↓)	A-OSE (↓)	mAP (↑)
5	0.04094	9300	55.73
10	0.02193	8234	56.34
15	0.01049	8088	56.65

Table 9: Increasing the margin Δ, improves the performance on known and unknown classes, concurring with our assumption that separation in the latent space is beneficial for ORE.

D. Varying the Temperature (T) in Eqn. 4

We fixed the temperature parameter (T) in Eqn. 4 to 1 in all the experiments. Softening the energies a bit more to $T = 2$, gives slight improvement in unknown detection, however increasing it further hurts as evident from Tab. 10.

T	WI(↓)	A-OSE(↓)	mAP(↑)
1	0.0219	8234	56.34
2	0.0214	8057	55.68
3	0.0411	11266	55.51
5	0.0836	12063	56.25
10	0.0835	12064	56.31

Table 10: There is a nice ballpark for temperature parameter between $T = 1$ and $T = 2$, which gives the optimal performance.

E. More Details on Contrastive Clustering

The motivation for using contrastive clustering to ensure separation in the latent space is two-fold: 1) it enables the model to cluster unknowns separately from known instances, thus boosting unknown identification; 2) it ensures instances of each class are well-separated from other classes, alleviating the forgetting issue.

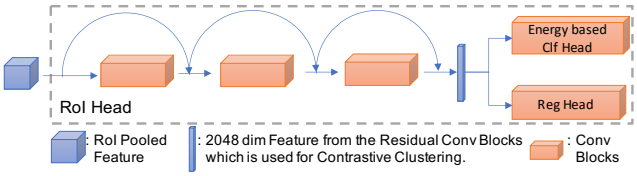


Figure 6: RoI head architecture, showing 2048-dim feature vector used for contrastive clustering.

The 2048-dim feature vector that comes out from residual blocks of RoI head (Fig 6) is contrastively clustered. The contrastive loss is added to the Faster R-CNN loss and the entire network is trained end-to-end. Thus all parts of the network before and including the residual block in the RoI head in the Faster R-CNN pipeline will get updated with the gradients from the contrastive clustering loss.

F. Further Implementation Details

We complete the discussion related to the implementation details, that we had in Sec. 5.2 here. We ran our experiments on a server with 8 Nvidia V100 GPUs with an effective batch size of 8. We use SGD with a learning rate of 0.01. Each task is learned for 8 epochs ($\sim$ 50k iterations). The queue size of the feature store is set to 20. We initiate clustering after 1k iterations and update the cluster prototypes after each 3k iterations with a momentum parameter of 0.99. Euclidean distance is used as the distance function $\mathcal{D}$ in Eqn. 1. The margin (Δ) is set as 10. For auto-labelling the unknowns in the RPN, we pick the top-1 background proposal, sorted by its objectness score. The temperature parameter in the energy based classification head is set to 1. The code is implemented in PyTorch [44] using Detectron 2 [63]. Reliability library [53] was used for modelling the energy distributions. We release all our codes publicly for foster reproducible research: https://github.com/JosephKJ/OWOD.

G. Related Work on Incremental Object Detection

The class-incremental object detection (iOD) setting considers classes to be observed incrementally over time and that the learner must adapt without retraining on old classes from scratch. The prevalent approaches [61, 28, 18, 7] use knowledge distillation [21] as a regularization measure to avoid forgetting old class information while training on new classes. Specifically, Shmelkov *et al*. [61] repurpose Fast R-CNN for incremental learning by distilling classification and regression outputs from a previous stage model. Beside distilling model outputs, Chen *et al*. [7] and Li *et al*. [28] also distilled the intermediate network features. Hao *et al*. [18] builds on Faster R-CNN and uses a student-teacher framework for RPN adaptation. Acharya *et al*. [1] proposes a replay mechanism for online detection. Recently, Peng *et al*. [45] introduces an adaptive distillation technique into Faster R-CNN. Their methodology is the current state-of-the-art in iOD. These methods cannot however work in an Open World environment, which is the focus of this work, and are unable to identify unknown objects.

H. Time and Storage Expense:

The training and inference of ORE takes an additional 0.1349 sec/iter and 0.009 sec/iter than standard Faster R-CNN. The storage expense for maintaining F_{Store} is negligible, and the exemplar memory (for $N_{ex} = 50$) takes approximately 34 MB.

I. Using Softmax based Unknown Identifier

We modified the unknown identification criteria to *max(softmax(logits))* $<$ t. For t = $\{0.3, 0.5, 0.7\}$: A-OSE, WI and mAP (mean and std-dev) are 11815 ± 352.13, 0.0436 ± 0.009 and 55.22 ± 0.02. This is inferior to ORE.

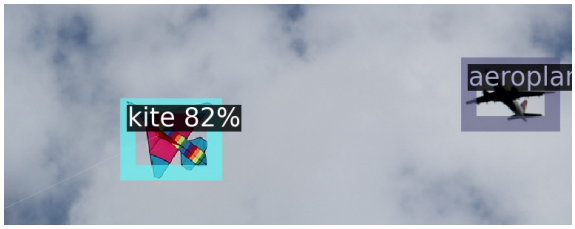
J. Qualitative Results

We show qualitative results of ORE in Fig. 8 through Fig. 13. We see that ORE is able to identify a variety of unknown instances and incrementally learn them, using the proposed contrastive clustering and energy-based unknown identification methodology. Sub-figure (a) in all these images shows the identified unknown instances along with the the other instances known to the detector. The corresponding sub-figure (b), shows the detections from the same detector after the new classes are incrementally added.

K. Discussion Regarding Failure Cases

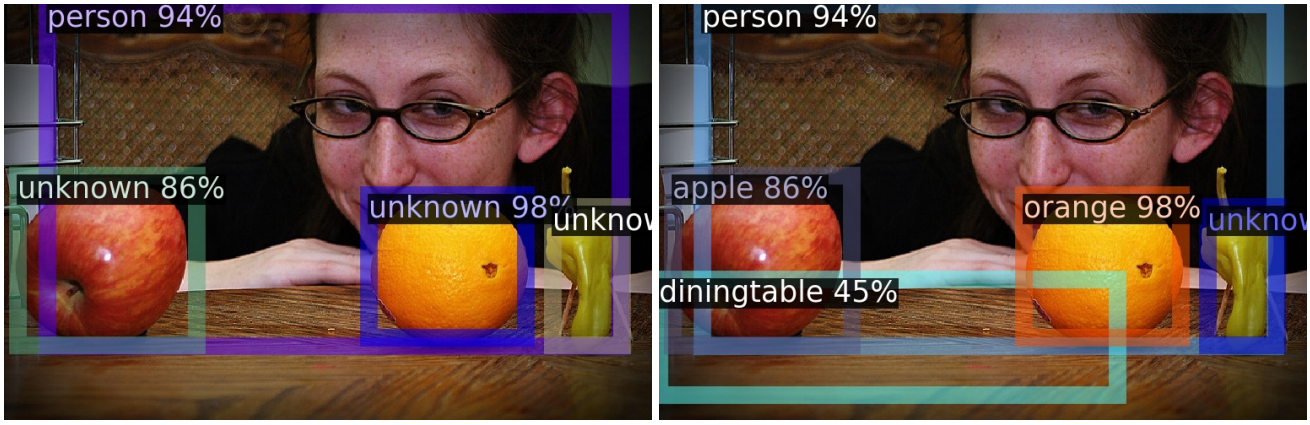
Occlusions and crowding of objects are cases where our method tends to get confused (`external-storage`, `walkman` and `bag` not detected as `unknown` in Figs. 11, 13). Difficult viewpoints (such as backside) also lead to some misclassifications (`giraffe`→`horse` in Figs. 4, 12). We have also noticed that detecting small *unknown* objects co-occurring with larger known objects is hard. As ORE is the first effort in this direction, we hope these identified shortcomings would be basis of further research.

(a)



(b)

Figure 7: ORE trained on just Task 1, successfully localises a `kite` as an unknown in sub-figure (a), while after learning about `kite` in Task 3, it incrementally learns to detect both `kite` and `aeroplane` in sub-figure (b).

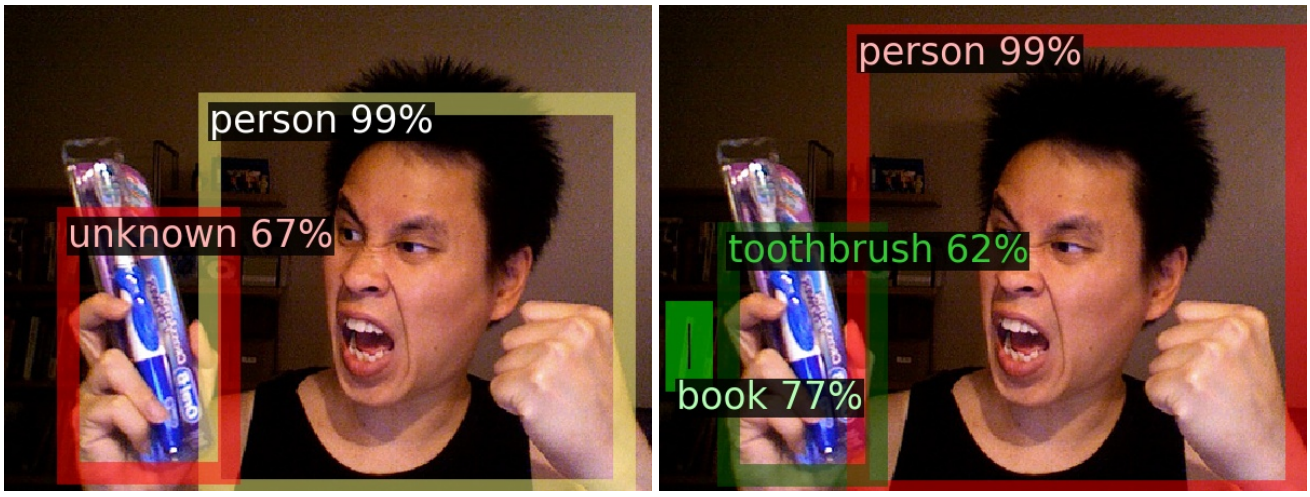


(a) (b)

Figure 8: The sub-figure (a) is the result produced by ORE after learning Task 2. As Task 3 classes like `apple` and `orange` has not been introduced, ORE identifies it and correctly labels them as `unknown`. After learning Task 3, these instances are labelled correctly in sub-figure (b). An unidentified class instance still remains, and ORE successfully detects it as `unknown`.

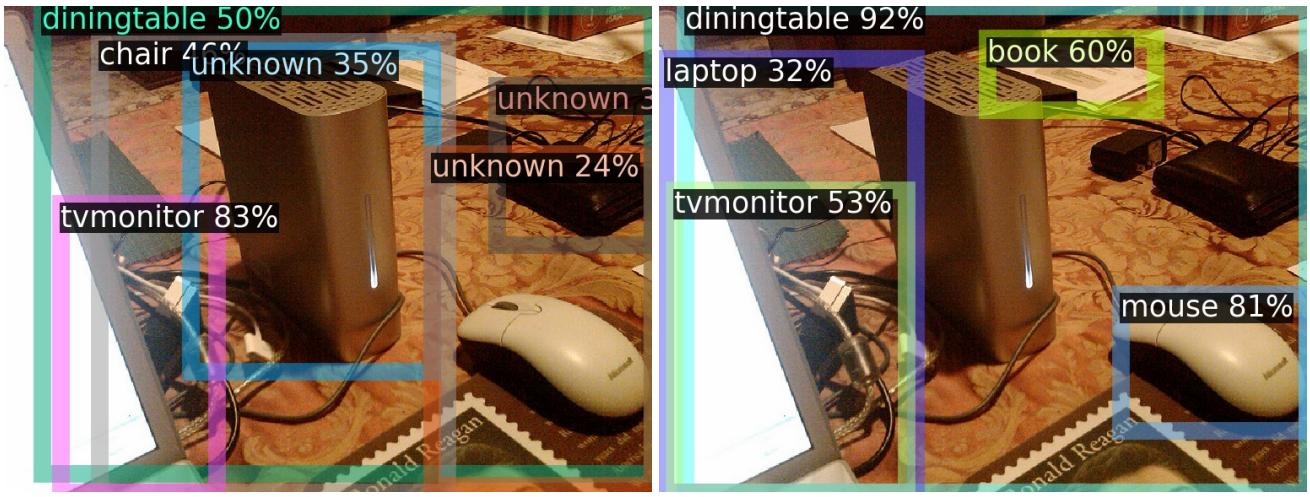
(a) (b)

Figure 9: The `clock` class is eventually learned as part of Task 4 (in sub-figure (b)), after being initially identified as `unknown` (in sub-figure (a)). ORE exhibits the true characteristics of an Open World detector, where it is able to incrementally learn an identified unknown.



(a) (b)

Figure 10: `toothbrush` and `book` are indoor objects introduced as part of Task 4. The detector trained till Task 3, identifies `toothbrush` as an unknown objects in sub-figure (a) and eventually learn it as part of Task 4, without forgetting how to identify `person` in sub-figure (b).

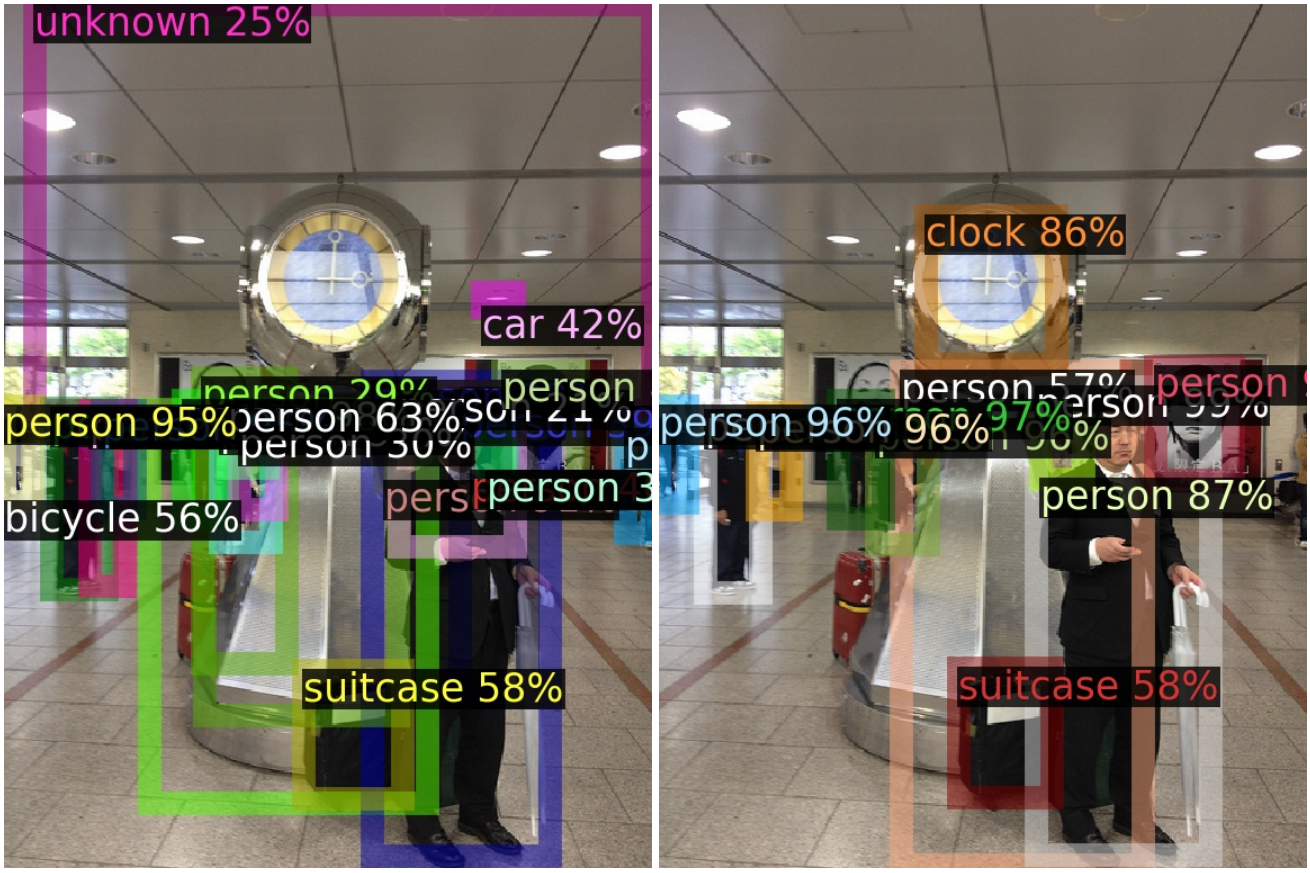


(a) (b)

Figure 11: Several items next to a `laptop` on top of a table are identified as `unknown`, after learning Task 1. `laptop`, `book` and `mouse` are introduced as part of Task 4, and hence are detected afterwords. `external-storage` and `walkman` (both are never introduced) were identified as `unknown` initially, but has not been detected after learning Task 4, and is one of the failure cases of ORE.

(a) (b)

Figure 12: `suitcase` which was identified as `unknown` is eventually learned in Task 2, along with a false positive detection of `chair`.



(a) (b)

Figure 13: In this highly cluttered scene, the `unknown` instance `clock` is identified, but is not localised well, after learning Task 2. After learning Task 4, ORE detects `clock`, along with reducing false positive detections of `car` and `bicycle`. The red `suitcase` is not labelled after learning either of the tasks, and hence is a failure case.