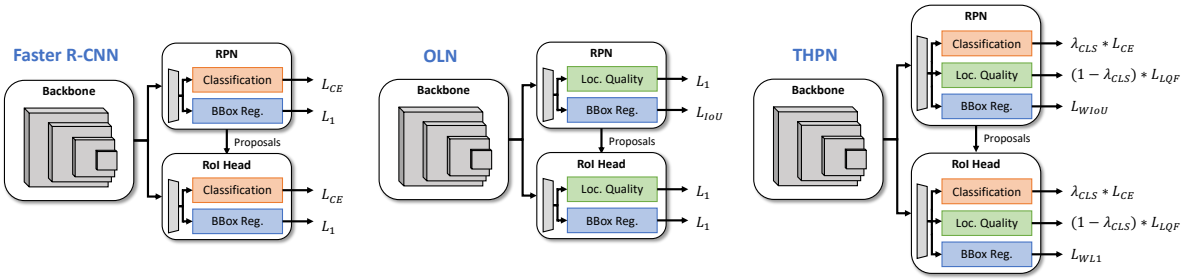

Figure 5. Differences between Faster R-CNN, OLN, and our hybrid THPN architecture.

Appendix

A. Overview of baseline methods

There are currently three types of object proposals methods: (1) learning-free approaches, (2) classification-based learning approaches, and (3) pure localization-based learning approaches. In this section, we outline the methodology for each and discuss their respective drawbacks. See Fig. 5 for a visual representation of the architectural differences between our approach and existing proposal networks.

Learning-free approaches. Early approaches for generating object proposals rely on hand-crafted features such as contrast and edges. Krähenbühl and Koltun [34] identify critical level sets in geodesic distance transforms computed over carefully placed seeds to generate proposals. Uijlings et al. [60] propose Selective Search, an algorithm that performs a graph-based segmentation and greedily merges superpixels based on color, texture, size, and shape similarity. Edge Boxes [75] is a box proposal algorithm that works by efficiently computing and finding enclosed edge contours in an image. Finally, Multiscale Combinatorial Grouping (MCG) [50] performs multi-scale segmentation and groups regions by efficiently exploring their combinatorial space. These methods were once the backbone of many early object detection systems such as Fast R-CNN [16], but have since been outclassed by learning-based approaches in terms of efficiency and recall performance.

Classification-based learning approaches. The seminal learning-based proposal solution is the Region Proposal Network (RPN) [51]. For a predefined set of anchor boxes, RPN learns to predict box coordinate regression deltas as well as an "objectness" score which indicates the likelihood that the box contains a foreground object of interest as opposed to background or an object from a class outside of the training distribution. While the single-stage RPN serves as an efficient benchmark solution for many state-of-the-art detection systems [6, 21, 51], we primarily focus on two-stage approaches in this work as they have been shown to have significantly better generalization abilities [29, 32, 64]. Thus, we consider a class-agnostic Faster R-CNN model as a stronger baseline, in which all annotations are treated as instances of the same class. The loss function for both stages of a class agnostic Faster R-CNN model (i.e., RPN and RoI Head) for an image is defined as:

$$L_{RPN}\big(\{c_i\},\{t_i\}\big) = \frac{1}{N_{cls}}\sum_i L_{BCE}(c_i, c_i^*) + \lambda\frac{1}{N_{reg}}\sum_i c_i^* L_1(t_i, t_i^*). \tag{4}$$

Here, i is the anchor index, c_i is the predicted probability of anchor i containing an object, and t_i are the predicted box regression deltas for anchor i. The ground truth labels for objectness and box regression are shown as c_i^* and t_i^*, respectively. The box regression head is trained with an L_1 loss. The objectness head of RPN is trained with a binary cross-entropy loss L_{BCE}, and a sampler is used to ensure 50% of boxes sampled during training are positive matches to some ground truth object.

Kim et al. [32] point out that framing object proposal as a discriminative task hinders generalization as it involves indiscriminate sampling of negative regions when training the objectness head(s). In other words, because we only have access to labels from a subset of object classes that exist in the world, the model overfits to the labeled object classes and treats all unlabeled objects as background.

Pure localization-based learning approach. Object Localization Network (OLN) [32] provides an alternative to classification-based approaches in an effort to resolve the aforementioned overfitting issue. OLN uses the same architecture

as Faster R-CNN, but it replaces the classification heads with localization quality prediction heads. OLN uses centerness [59] and IoU score [28] as substitutes for objectness in the OLN-RPN and OLN-Box stages, respectively. Thus, OLN is a pure localization-based proposal network trained with the following loss function:

$$L_{OLN}(\{q_j\},\{t_i\}) = \frac{1}{N_{lq}}\sum_j L_1(q_j, q_j^*) + \lambda\frac{1}{N_{reg}}\sum_i L_1(t_i, t_i^*). \quad (5)$$

Here, q_j and q_j^* are the predicted and target localization quality scores, respectively. Note that we use a separate index j because the set of sampled anchors for the LQ head can be different than the set of sampled anchors for the BOX head. Because we are not training a discriminative classification task, we only need to sample positively matched anchors to train the LQ head. Intuitively, if a model can accurately predict its overlap $q_j \in [0, 1]$ with a ground truth object, we can effectively treat q_j as a notion of objectness. This re-framing of the sparse classification problem allows the model to be less biased towards the specific classes that it is trained on.

While OLN resolves the explicit bias resulting from learning to classify all unlabeled regions as background, we posit that it still suffers from implicit bias because it still only learns from ID instances. Adding to the bias problem is the fact that there exists a natural class imbalance issue in natural data. While OLN's generalization benefits over Faster R-CNN is shown to be significant when it is trained on a diverse and representative class set such as PASCAL VOC [12], we hypothesize that it will struggle when faced with more challenging tasks with fewer ID classes and fewer labeled instances.

B. Learning-free baseline comparison

	ALL	
Method	AR10	AR100
Geodesic [34]	4.0	18.0
Sel. Search [60]	5.2	16.3
EdgeBoxes [75]	7.4	17.8
MCG [50]	10.1	24.6
THPN ($\lambda_{CLS} = 0.50$)	**41.0**	**59.9**

Table 6. ALL recall comparison with learning-free baselines on the COCO dataset. The THPN (trained on all COCO classes) significantly outperforms these approaches. Results for baselines are borrowed from [49].

While learning-free approaches to object proposal have been largely supplanted by deep learning-based methods, we feel that they are still worth worth comparing against. For this test, we compare recall on all classes in the COCO validation set. For this experiment, we use a THPN with $\lambda_{CLS} = 0.50$ to promote the optimal ID recall. Tab. 6 shows that THPN outclasses all learning-free baselines by a significant margin.

C. Closed-set performance

THPN's design is primarily suited for handling open-set tasks that assume the presence of OOD objects of interest. However, we notice that a THPN with a larger λ_{CLS} is capable of superior ID performance to pure classification-based models like Faster R-CNN. For completeness, we test our model's performance in closed-set tasks. The results of this test are in Tab. 7. We train each model on all classes of COCO or ShipRSImageNet (except *docks*) and test on the complete validation sets. Note that we do not use self-training or crop & zoom augmentations here to test the impact of the architectural differences only. On the large-scale COCO dataset, Faster R-CNN outperforms OLN. However, THPN with $\lambda_{CLS} \geq 0.10$ beats both baselines. Setting $\lambda_{CLS} = 0.50$ is the best in this case, beating Faster R-CNN by +2.4 AUC. On the smaller ships data, OLN is superior to Faster R-CNN, and THPN with $\lambda_{CLS} \leq 0.25$ outperforms OLN by +0.8 AUC.

D. Data augmentation and training schedule

Tab. 8 shows the effect of various data augmentations and training schedules on an OLN model trained on the VOC split. In this work, we borrow transform implementations from mmdetection [7]. We find that the crop & zoom augmentation is the most effective for improving OOD generalization of an OLN-style of proposal network. However, all augmentations come at a slight cost of ID recall. We also find that using a 2x training schedule (16 epochs) is beneficial only if using the additional strong augmentations.

Dataset	Model	Images	Instances	Test ALL AUC	AR10	AR100	AR1000
COCO	Faster R-CNN	117k	860k	42.9	38.1	57.1	63.1
	OLN	117k	860k	39.6	29.8	54.8	64.4
	THPN ($\lambda_{CLS} = 0$)	117k	860k	39.6	29.8	54.8	64.4
	THPN ($\lambda_{CLS} = 0.10$)	117k	860k	43.1	37.4	57.5	65.0
	THPN ($\lambda_{CLS} = 0.25$)	117k	860k	44.7	40.4	59.1	65.6
	THPN ($\lambda_{CLS} = 0.50$)	117k	860k	**45.3**	**41.0**	**59.9**	**66.2**
	THPN ($\lambda_{CLS} = 0.75$)	117k	860k	45.1	40.6	59.7	66.2
	THPN ($\lambda_{CLS} = 0.90$)	117k	860k	44.8	40.1	59.3	66.2
	THPN ($\lambda_{CLS} = 1$)	117k	860k	44.7	39.9	59.2	66.0
Ships	Faster R-CNN	2.2k	10k	50.2	51.5	63.1	68.4
	OLN	2.2k	10k	51.4	51.5	65.7	70.3
	THPN ($\lambda_{CLS} = 0$)	2.2k	10k	51.6	51.7	**66.0**	**70.3**
	THPN ($\lambda_{CLS} = 0.10$)	2.2k	10k	51.6	52.7	65.6	69.6
	THPN ($\lambda_{CLS} = 0.25$)	2.2k	10k	**51.8**	**53.1**	65.8	69.7
	THPN ($\lambda_{CLS} = 0.50$)	2.2k	10k	51.1	52.6	64.6	68.9
	THPN ($\lambda_{CLS} = 0.75$)	2.2k	10k	51.4	52.8	65.2	69.4
	THPN ($\lambda_{CLS} = 0.90$)	2.2k	10k	50.9	52.4	64.3	69.3
	THPN ($\lambda_{CLS} = 1$)	2.2k	10k	51.0	52.3	64.9	68.8

Table 7. Results on closed-set tasks. Here, we assume all instances of all classes of interest are labeled.

Augmentation	Epochs	OOD-AUC	ID-AUC	ALL-AUC
None	8	24.8	**44.8**	35.5
Crop & Zoom	8	25.5	43.0	34.8
Discrete Rotate	8	24.8	40.3	33.0
Random Affine	8	23.5	42.3	33.6
Photometric Distortion	8	25.0	44.4	35.4
Crop & Zoom + Photometric Distortion	8	25.5	42.0	34.2
None	16	24.8	44.8	35.5
Crop & Zoom	16	**26.1**	44.2	**35.8**
Discrete Rotate	16	25.4	41.8	34.0
Random Affine	16	24.0	43.7	34.5
Crop & Zoom + Photometric Distortion	16	26.0	43.7	35.4

Table 8. Effect of augmentation and training schedule on an OLN model trained on the VOC split.

Split	Model	Images / Instances	OOD AUC	OOD AR10	OOD AR100	OOD AR1k	ID AUC	ID AR10	ID AR100	ID AR1k	ALL AUC	ALL AR10	ALL AR100	ALL AR1k
Military	Faster R-CNN	1.5k / 4.7k	11.8	11.8	12.2	24.3	65.6	75.6	79.4	81.3	33.3	37.1	39.1	47.1
	OLN	1.5k / 4.7k	23.0	23.4	28.1	35.8	69.0	79.2	84.4	85.4	41.3	45.2	50.6	55.6
	THPN ($\lambda_{CLS} = 0$)	1.5k / 6.1k	**29.0**	**28.4**	**37.1**	**40.7**	68.8	78.3	84.2	85.1	**44.7**	**47.7**	**56.0**	**58.5**
	THPN ($\lambda_{CLS} = 0.10$)	1.5k / 6.1k	26.6	27.0	33.6	37.8	**69.3**	**79.4**	**84.7**	**85.4**	43.6	47.5	54.1	56.9
	THPN ($\lambda_{CLS} = 0.25$)	1.5k / 6.1k	22.4	23.1	27.3	33.8	69.0	79.3	84.1	84.9	40.9	45.2	50.0	54.3
	THPN ($\lambda_{CLS} = 0.50$)	1.5k / 6.1k	16.1	16.8	19.2	25.8	68.3	78.7	83.1	84.4	36.9	41.3	44.7	49.2
Civilian	Faster R-CNN	1.0k / 4.4k	16.0	16.8	17.8	29.1	33.4	29.5	44.3	51.2	24.6	22.6	31.0	40.1
	OLN	1.0k / 4.4k	38.7	38.4	49.2	56.6	35.6	30.7	47.9	54.3	36.9	33.8	48.5	55.5
	THPN ($\lambda_{CLS} = 0$)	1.0k / 5.7k	**49.8**	**50.4**	**63.5**	**68.3**	**36.5**	30.3	**49.1**	**56.7**	**42.8**	**39.3**	**56.2**	**62.5**
	THPN ($\lambda_{CLS} = 0.10$)	1.0k / 5.7k	46.3	47.2	58.4	64.2	36.4	31.4	48.5	55.0	41.0	38.2	53.4	59.6
	THPN ($\lambda_{CLS} = 0.25$)	1.0k / 5.7k	33.9	34.4	41.6	52.5	35.8	**31.5**	47.6	54.0	34.6	32.0	44.5	53.3
	THPN ($\lambda_{CLS} = 0.50$)	1.0k / 5.7k	24.1	22.8	28.7	42.8	35.5	31.1	46.9	54.1	29.5	26.2	37.7	48.5

Table 9. Full results on the *ships* challenge.

E. Full ship detection results

For the purposes of space efficiency, we show a summary of the results from the *ships* challenge in Sec. 5.4 of the main text. In this section, we provide the full results of this experiment in Tab. 9. In both splits, THPN with $\lambda_{CLS} = 0$ is vastly

Split	Model	Images / Instances	OOD AUC	OOD AR10	OOD AR100	OOD AR1k	ID AUC	ID AR10	ID AR100	ID AR1k	ALL AUC	ALL AR10	ALL AR100	ALL AR1k
COCO40	Faster R-CNN	104k / 623k	26.6	17.5	36.0	51.4	44.4	41.4	58.3	63.2	39.0	33.8	51.7	60.0
	OLN	104k / 623k	33.1	25.8	44.8	54.6	42.1	34.6	57.2	65.0	38.9	30.5	53.3	62.2
	THPN ($\lambda_{CLS} = 0$)	104k / 810k	34.1	28.0	45.6	55.2	41.6	34.6	56.2	64.0	38.9	31.1	52.8	61.6
	THPN ($\lambda_{CLS} = 0.10$)	104k / 810k	**34.8**	**29.8**	**46.0**	55.3	44.0	39.6	58.1	64.6	40.7	35.1	54.3	62.0
	THPN ($\boldsymbol{\lambda_{CLS} = 0.25}$)	104k / 810k	34.5	29.2	45.6	**55.4**	45.0	41.5	59.0	**65.0**	**41.5**	**36.5**	**54.9**	**62.4**
	THPN ($\lambda_{CLS} = 0.50$)	104k / 810k	33.6	28.0	44.5	55.3	**45.1**	**41.7**	**60.7**	64.7	34.4	29.9	44.8	55.1
VOC	Faster R-CNN	95k / 493k	19.3	11.6	25.1	42.4	46.7	45.1	60.7	64.7	34.4	29.9	44.8	55.1
	OLN	95k / 493k	24.8	18.4	33.2	45.0	44.8	40.1	59.3	66.1	35.5	29.1	47.5	56.9
	THPN ($\lambda_{CLS} = 0$)	95k / 641k	27.7	21.3	36.9	48.0	44.8	39.9	59.5	66.0	36.7	30.1	49.3	58.3
	THPN ($\lambda_{CLS} = 0.10$)	95k / 641k	**27.9**	**22.0**	**37.1**	**48.0**	46.8	44.2	60.9	66.5	38.0	32.9	50.2	58.5
	THPN ($\boldsymbol{\lambda_{CLS} = 0.25}$)	95k / 641k	27.5	21.4	36.6	48.0	47.6	45.8	61.8	66.8	**38.4**	**33.7**	**50.5**	58.8
	THPN ($\lambda_{CLS} = 0.50$)	95k / 641k	26.4	19.8	35.0	47.8	**47.8**	**46.1**	**62.1**	**67.1**	38.0	33.5	50.0	**58.9**
VOC5	Faster R-CNN	74k / 357k	16.3	9.8	20.7	38.1	48.1	47.6	62.2	65.6	29.1	24.8	37.4	49.6
	OLN	74k / 357k	20.3	14.1	26.9	40.1	47.6	45.2	61.7	67.8	31.0	25.7	40.8	51.6
	THPN ($\lambda_{CLS} = 0$)	74k / 465k	23.5	17.1	31.3	43.9	46.8	43.9	61.1	67.0	32.6	26.7	43.2	53.7
	THPN ($\lambda_{CLS} = 0.10$)	74k / 465k	**23.7**	**17.6**	**31.5**	**43.9**	48.3	47.1	62.4	67.4	33.3	28.4	43.8	53.7
	THPN ($\boldsymbol{\lambda_{CLS} = 0.25}$)	74k / 465k	23.6	17.4	31.1	43.9	49.0	48.4	63.1	67.7	**33.5**	**28.9**	**43.9**	**53.8**
	THPN ($\lambda_{CLS} = 0.50$)	74k / 465k	22.7	16.1	30.0	43.9	**49.3**	**48.7**	**63.5**	**67.8**	33.2	28.5	43.4	53.8
Animal	Faster R-CNN	24k / 63k	11.5	6.0	13.5	31.3	53.9	58.9	67.1	69.4	14.6	9.8	17.5	34.1
	OLN	24k / 63k	13.3	8.2	16.4	31.5	55.8	59.7	69.7	73.2	16.4	11.9	20.3	34.6
	THPN ($\lambda_{CLS} = 0$)	24k / 81k	16.9	9.9	22.9	**36.7**	55.5	59.0	69.7	73.0	19.7	13.4	26.3	39.3
	THPN ($\lambda_{CLS} = 0.10$)	24k / 81k	**17.0**	**10.4**	**22.9**	36.6	56.1	60.6	69.9	73.0	19.8	13.9	26.3	39.3
	THPN ($\boldsymbol{\lambda_{CLS} = 0.25}$)	24k / 81k	17.0	10.3	22.8	36.6	56.6	61.5	70.5	73.2	**19.8**	**14.0**	**26.3**	**39.3**
	THPN ($\lambda_{CLS} = 0.50$)	24k / 81k	15.9	9.3	20.7	36.6	**56.8**	**61.9**	**70.8**	**73.3**	18.9	13.1	24.4	39.3

Table 10. Results on the *training class diversity* challenge when using a THPN trained with $\lambda_{CLS} = 0.10$. The listed λ_{CLS} in the table is the value used during inference-time.

Split	Model	Images / Instances	OOD AUC	OOD AR10	OOD AR100	OOD AR1k	ID AUC	ID AR10	ID AR100	ID AR1k	ALL AUC	ALL AR10	ALL AR100	ALL AR1k
VOC (50%)	Faster R-CNN	75k / 246k	18.7	11.7	24.1	40.9	44.8	42.7	58.5	63.1	33.1	28.5	43.2	53.6
	OLN	75k / 246k	23.8	17.7	31.7	43.8	44.4	39.5	58.8	65.7	34.9	28.5	46.7	56.3
	THPN ($\lambda_{CLS} = 0$)	75k / 320k	25.7	19.4	**34.2**	**45.9**	44.7	40.9	58.6	65.4	35.9	30.1	47.6	57.1
	THPN ($\lambda_{CLS} = 0.10$)	75k / 320k	**25.7**	**19.6**	34.0	45.9	46.1	44.1	59.7	65.7	36.7	32.0	48.1	57.2
	THPN ($\boldsymbol{\lambda_{CLS} = 0.25}$)	75k / 320k	25.1	18.5	33.3	45.8	46.8	**45.3**	60.5	66.0	**36.9**	**32.4**	**48.3**	**57.4**
	THPN ($\lambda_{CLS} = 0.50$)	75k / 320k	23.7	16.3	31.4	45.7	**46.8**	44.9	**60.8**	**66.2**	36.3	31.5	47.6	57.4
VOC (25%)	Faster R-CNN	56k / 123k	17.9	11.2	22.9	39.2	42.7	40.1	55.8	60.9	31.6	27.0	41.1	51.6
	OLN	56k / 123k	21.9	16.6	28.8	40.7	43.2	38.3	57.1	64.1	33.4	27.5	44.5	54.0
	THPN ($\lambda_{CLS} = 0$)	56k / 160k	**24.3**	**17.9**	**32.3**	**44.8**	43.6	39.6	57.2	64.1	34.6	28.8	46.0	55.8
	THPN ($\lambda_{CLS} = 0.10$)	56k / 160k	24.2	17.9	32.1	44.8	44.7	42.4	58.0	64.2	35.3	30.5	46.4	55.9
	THPN ($\boldsymbol{\lambda_{CLS} = 0.25}$)	56k / 160k	23.7	17.0	31.4	44.7	45.2	**43.3**	58.5	64.3	**35.4**	**30.7**	**46.4**	**56.0**
	THPN ($\lambda_{CLS} = 0.50$)	56k / 160k	22.6	15.2	29.9	44.6	**45.3**	43.1	**58.8**	**64.5**	35.0	30.0	45.8	56.0
VOC (10%)	Faster R-CNN	33k / 49k	16.2	10.4	20.5	35.8	39.5	36.2	51.8	57.8	29.1	24.5	37.9	48.4
	OLN	33k / 49k	19.8	15.2	25.7	37.3	40.8	36.3	53.6	61.0	31.3	26.0	41.3	50.8
	THPN ($\lambda_{CLS} = 0$)	33k / 64k	22.9	16.9	**30.3**	**42.4**	41.5	37.3	54.5	61.8	33.0	27.2	43.7	53.5
	THPN ($\lambda_{CLS} = 0.10$)	33k / 64k	**23.0**	**17.2**	30.2	42.4	42.3	39.5	55.0	61.9	33.5	28.6	43.9	53.6
	THPN ($\boldsymbol{\lambda_{CLS} = 0.25}$)	33k / 64k	22.6	16.4	29.9	42.4	42.8	**40.4**	55.5	62.1	**33.6**	**28.9**	**44.1**	53.7
	THPN ($\lambda_{CLS} = 0.50$)	33k / 64k	21.9	15.0	29.1	42.5	**43.0**	40.3	**55.9**	**62.2**	33.4	28.4	44.0	**53.8**

Table 11. Results on the *semi-supervised* challenge when using a THPN trained with $\lambda_{CLS} = 0.10$. The listed λ_{CLS} in the table is the value used during inference-time.

superior to the baselines in terms of OOD recall (regardless of k in AR@k). Because OLN also effectively has $\lambda_{CLS} = 0$, it is clear that our self-training procedure is very effective in this domain. This is likely due to the fact that there are relatively few labeled training instances compared to COCO. Given the correct setting of λ_{CLS}, THPN can also slightly improve the ID recall on both splits.

F. Inference-time dynamic λ_{CLS}

In all other experiments, the λ_{CLS} associated with the THPN models is consistent during both training (i.e., loss weighting and score weighting during pseudo-label generation) and inference (score weighting). For the sake of completeness, we include results where we use one λ_{CLS} for training, and alter it during inference. This implementation of THPN makes the model truly inference-time dynamic. We evaluate on the *training class diversity* and *semi-supervised* challenges using $\lambda_{CLS} = 0.10$ during training. The results of these experiments are in Table 10 and Table 11, respectively. Overall, we find that this inference-time-dynamic variant is capable of approaching the performance of the static-trained variant in terms of

ALL recall. However, it achieves this by creating a more optimal balance between OOD recall and ID recall, rather than improving either one individually. The dynamic variants have much less deviation in performance between λ_{CLS} values. Interestingly, the most optimal setting (in terms of ALL recall) for a dynamic THPN using $\lambda_{CLS} = 0.10$ during training is to use $\lambda_{CLS} = 0.25$ during inference. Regardless, these experiments confirm that THPN can be treated as dynamic at inference-time to good effect.

G. Pseudo-label visualizations

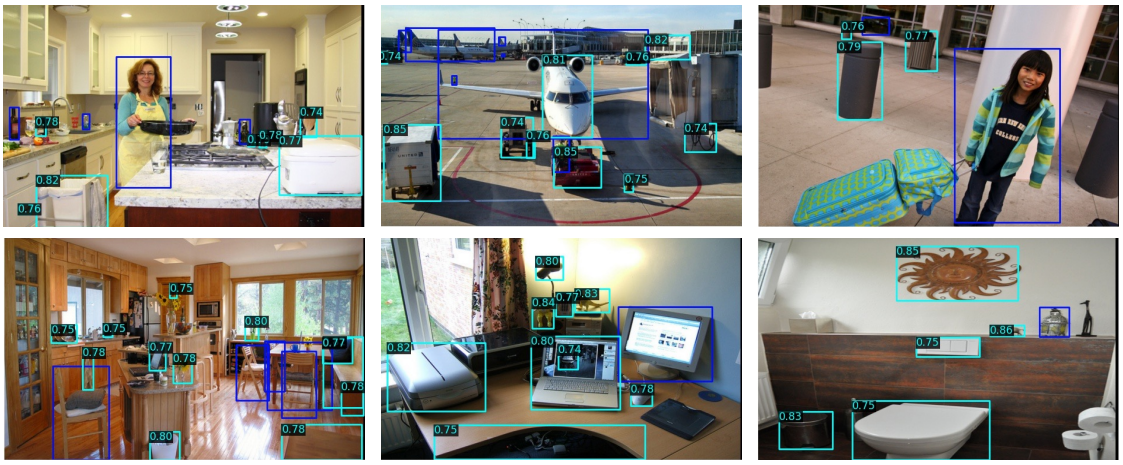

Figure 6. Visualization of pseudo-label boxes (cyan) alongside the original ground truth boxes (blue) on various training images. Here, we consider the VOC split and the pseudo-labels are from a THPN ($\lambda_{CLS} = 0.10$) model with p=30%.

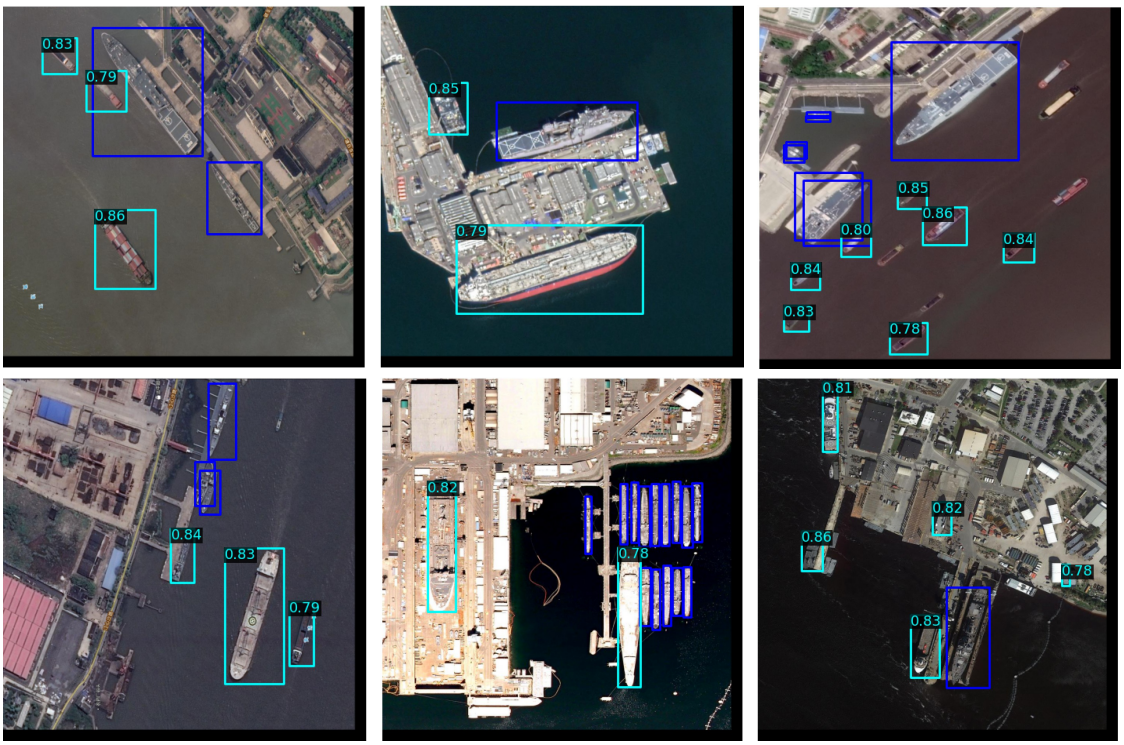

Figure 7. Visualization of pseudo-label boxes (cyan) alongside the original ground truth boxes (blue) on various training images. Here, we consider the Warship split and the pseudo-labels are from a THPN ($\lambda_{CLS} = 0$) model with p=30%.

H. Prediction visualizations

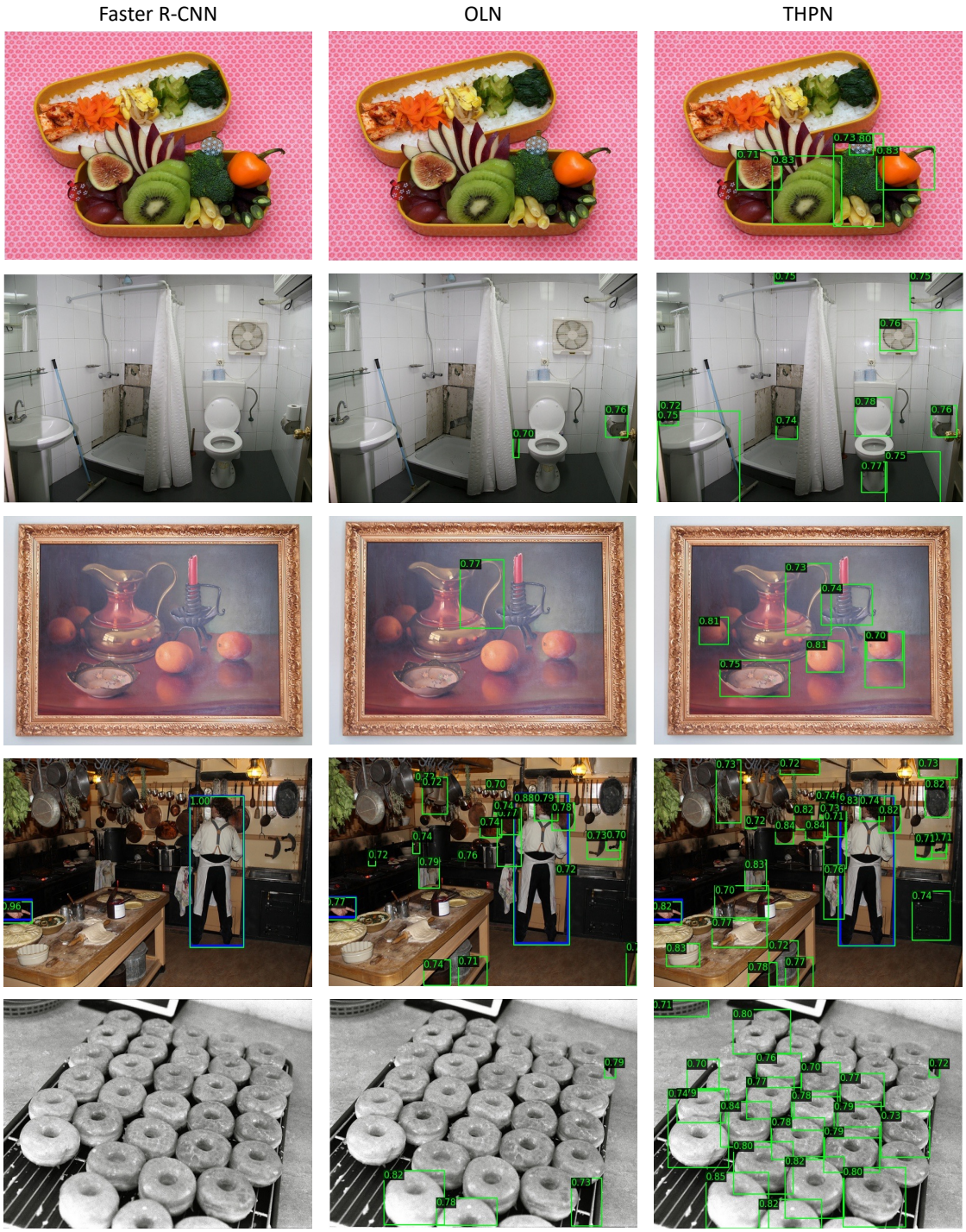

Figure 8. Visualization of predicted proposals (green) alongside the ground truth boxes (blue) on various COCO validation images. All models are trained on the VOC5 split (THPN is trained with $\lambda_{CLS} = 0$). In general, THPN is able to detect many OOD objects that Faster R-CNN and OLN misses. We find that a common failure mode of localization-based models is proposing parts of a larger object (e.g., a person's hand, the leg of a chair, etc.).

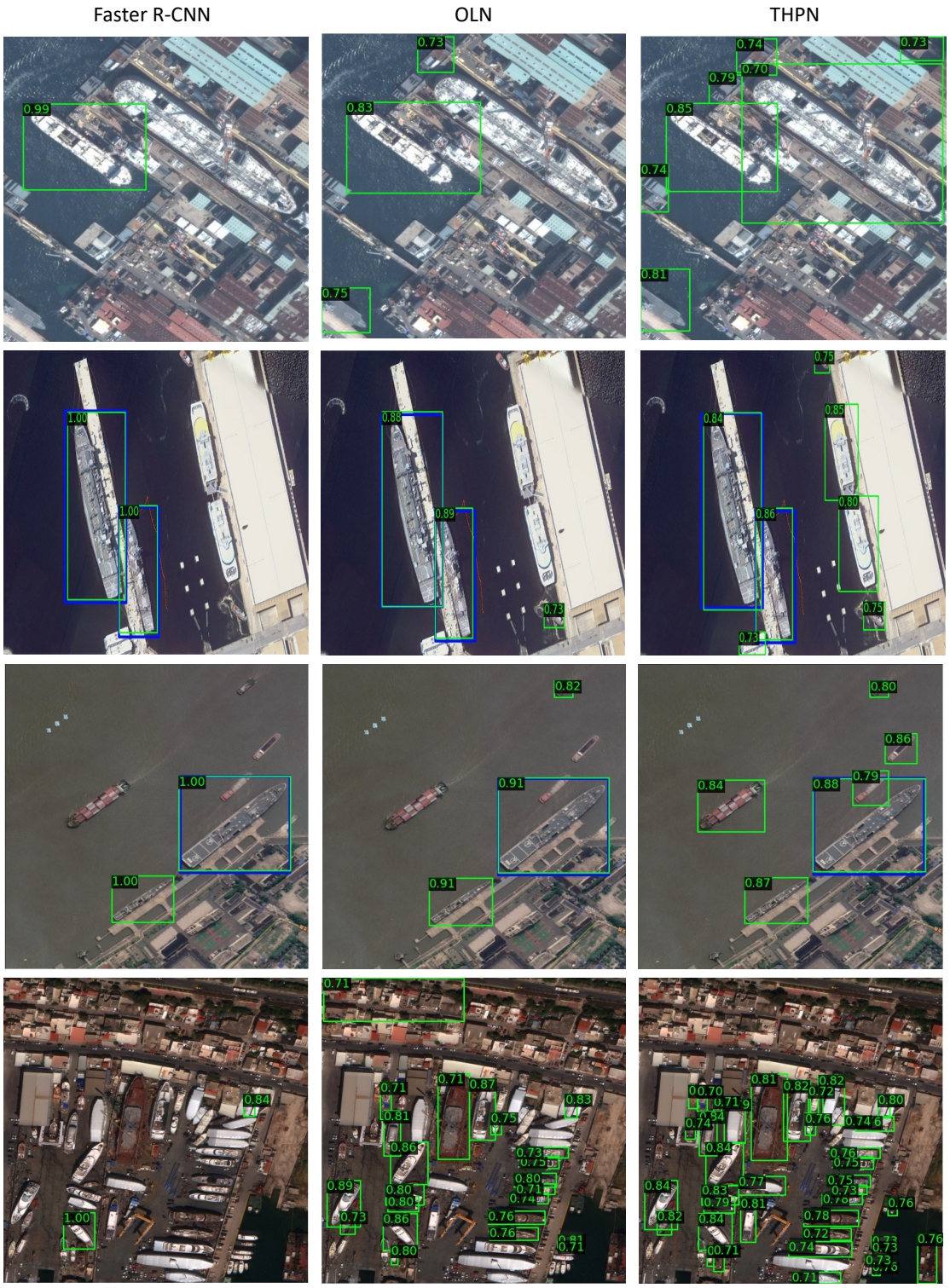

Figure 9. Visualization of predicted proposals (green) alongside the ground truth boxes (blue) on various ShipRSImageNet validation images. All models are trained on the Military split (THPN is trained with $\lambda_{CLS} = 0$). In general, THPN is able to detect many more OOD ships than Faster R-CNN and OLN while maintaining performance on the ID military ships. We find that a common failure mode of THPN on this dataset is to localize large defined objects on shore (e.g., storage containers, docks, etc.).

I. Training class diversity challenge splits

COCO Class	Super Category	COCO40	VOC	VOC5	Animal
person	person	✓	✓	✓	
bicycle	vehicle	✓	✓	✓	
car	vehicle	✓	✓	✓	
motorcycle	vehicle	✓	✓		
airplane	vehicle	✓	✓		
bus	vehicle	✓	✓		
train	vehicle	✓	✓		
truck	vehicle				
boat	vehicle	✓	✓		
traffic light	outdoor	✓			
fire hydrant	outdoor				
stop sign	outdoor	✓			
parking meter	outdoor				
bench	outdoor	✓			
bird	animal	✓	✓		✓
cat	animal	✓	✓		✓
dog	animal	✓	✓	✓	✓
horse	animal	✓	✓		✓
sheep	animal	✓	✓		✓
cow	animal	✓	✓		✓
elephant	animal				✓
bear	animal				✓
zebra	animal				✓
giraffe	animal				✓
backpack	accessory	✓			
umbrella	accessory				
handbag	accessory	✓			
tie	accessory				
suitcase	accessory				
frisbee	sports				
skis	sports	✓			
snowboard	sports				
sports ball	sports	✓			
kite	sports				
baseball bat	sports				
baseball glove	sports				
skateboard	sports	✓			
surfboard	sports	✓			
tennis racket	sports				
bottle	kitchen	✓	✓		
wine glass	kitchen				
cup	kitchen				
fork	kitchen	✓			
knife	kitchen				
spoon	kitchen				
bowl	kitchen	✓			
banana	food				
apple	food	✓			
sandwich	food				
orange	food				
broccoli	food				
carrot	food				
hot dog	food				
pizza	food	✓			
donut	food				
cake	food				
chair	furniture	✓	✓	✓	
couch	furniture	✓	✓		
potted plant	furniture	✓	✓		
bed	furniture				
dining table	furniture	✓	✓		
toilet	furniture	✓			
tv	electronic	✓	✓		
laptop	electronic	✓			
mouse	electronic				
remote	electronic	✓			
keyboard	electronic				
cell phone	electronic				
microwave	appliance				
oven	appliance	✓			
toaster	appliance				
sink	appliance	✓			
refrigerator	appliance	✓			
book	indoor				
clock	indoor				
vase	indoor				
scissors	indoor				
teddy bear	indoor				
hair drier	indoor				
toothbrush	indoor	✓			
% Train Set		72.5	57.3	41.6	7.3
% Test Set		72.2	57.0	41.5	7.3

Table 12. Training splits for the *training class diversity* challenge.

J. Frequently Asked Questions (FAQ)

- Why do we call this method "for the Open World"?
  - According to the terminology in the literature, "open-set recognition" refers to the task of dealing with OOD inputs [55], while "open-world recognition" refers to the task of detecting OOD inputs and incrementally learning them. In this work, we follow existing related work [32, 54] and refer to "open-world" more generally as any setting or environment that contains OOD instances in the test set. Importantly, THPN would be very useful to incorporate into both open-set object detection models *and* open-world object detection models.
- Why do we not measure Average Precision?
  - Computing precision for any class of object involves finding false positive predictions. In the case of class-agnostic open-set object detection, while we have labels for some of the OOD objects in the test images (e.g., the non-VOC COCO classes), this does not encapsulate *all* OOD objects in the test images. For example, "snowmobile" is not a COCO class, however there exists images with snowmobile instances in the validation set. It would be unfair to mark an accurate detection of an unlabeled snowmobile instance as a false positive. Therefore, we only consider recall. This is common practice in this field of study [32, 33, 54]. Furthermore, a proposal network's job is to maximize recall on all objects. In a full object detection model, there is a second stage which makes the final prediction refinement (e.g., R-CNN classification head).
- How is AR-AUC computed?
  - The most common metric for performance for proposal networks is Average Recall @ k detections per image (AR@k). The most common operating point to consider is AR@100, meaning the average recall when allowing 100 predictions per image. However, for some applications we may care about other operating points (e.g., k=10, k=1000, etc.). Therefore, a reasonable summary statistic to use is AUC introduced by Kim et al. [32] which computes the area under the curve of AR@k for k=$\{10, 30, 50, 100, 300, 500, 1000\}$.
- Does THPN's self-training require additional data that the baselines do not have access to?
  - As mentioned in Section 5, during training, not only do we assume access to *only* ID labels of the ID classes in the training set, we also ensure that THPN is only ever exposed to images that contain at least one ID label during training and pseudo-label generation. Thus, our implementation of THPN does not use any unlabeled training images, just like any non-self-trained baseline. While our implementation of THPN does not use auxiliary images for self-training for the sake of fairness, it is certainly possible to do so. This can be seen as an additional degree of flexibility of our method (as self-training can seamlessly ingest unlabeled data). We save investigating this for future work.
- Why is classification-based objectness beneficial for ID detection and localization-based objectness beneficial for OOD detection?
  - Due to space constraints, we could not include all details about classification-based objectness and localization-based objectness in the main paper. However, we detail these methods in Appendix A. Essentially, the difference in behavior comes down to the fact that classification-based models rely on discriminative learning which forces the model to sample negative regions that do not contain an ID object. The problem is, these negative regions often contain OOD objects. Therefore, a classification-based model explicitly treats OOD objects as *background*, meaning they are not detected during inference. Localization-based models instead frame objectness as the localization quality (e.g., centerness [59] or IoU [28]) between a given region and any ground truth box. Critically, learning localization-quality is not discriminative, so it does not require explicitly sampling negative regions. Thus, this approach is inherently more likely to generalize to OOD objects [32]. However, what is not shown in the OLN paper [32] is the fact that this improved OOD recall comes at the cost of reduced ID recall. Our solution (THPN) offers a tunable, flexible model that can achieve the best of both worlds by learning and using both objectness representations with a weight λ_{CLS}.
- Why do we not compare THPN to Open World Object Detection (OWOD) methods?

  - While it may seem reasonable to compare THPN to the increasing number of OWOD models in the literature [19, 29, 66, 70, 74], these models are for two very different tasks. OWOD models are full detection models that attempt to incrementally learn novel classes using a human-in-the-loop. THPN, on the other hand, is a class-agnostic proposal network. While THPN has the potential to greatly enhance the novel object detection capabilities as a component any OWOD model (i.e., by replacing RPN with THPN), we feel this investigation is beyond the scope of this paper, so we save it for future work.

- Can THPN made to be dynamic at inference time instead of only tunable at training time?
  - In our implementation of THPN, we assume that the λ_{CLS} used during training is the same λ_{CLS} used during inference. In this way, THPN is not necessarily meant to be tunable post-training. However, we also investigate what happens when we train with a moderate $\lambda_{CLS} = 0.10$, and modulate λ_{CLS} during inference. Results from this are in Appendix F. Overall, we find that while the results are not quite as good as when we modulate λ_{CLS} during training, an inference-time-dynamic variant of THPN is still quite effective. Either variant outperforms all baselines.

- What are the potential negative societal impacts?
  - Powerful object detection algorithms enable a huge range of potential automated applications. While many of these are righteous technologies (e.g., autonomous vehicles, more effective search and rescue, robots for the disabled, etc.), there are of course several nefarious applications such as aggressive surveillance or weapon systems. Regardless, due to the ubiquity of these models in the public domain, we argue that making these models as robust and trustworthy as possible is overall a noble and important endeavor.