SimLTD: Simple Supervised and Semi-Supervised Long-Tailed Object Detection

Phi Vu Tran
LexisNexis Risk Solutions

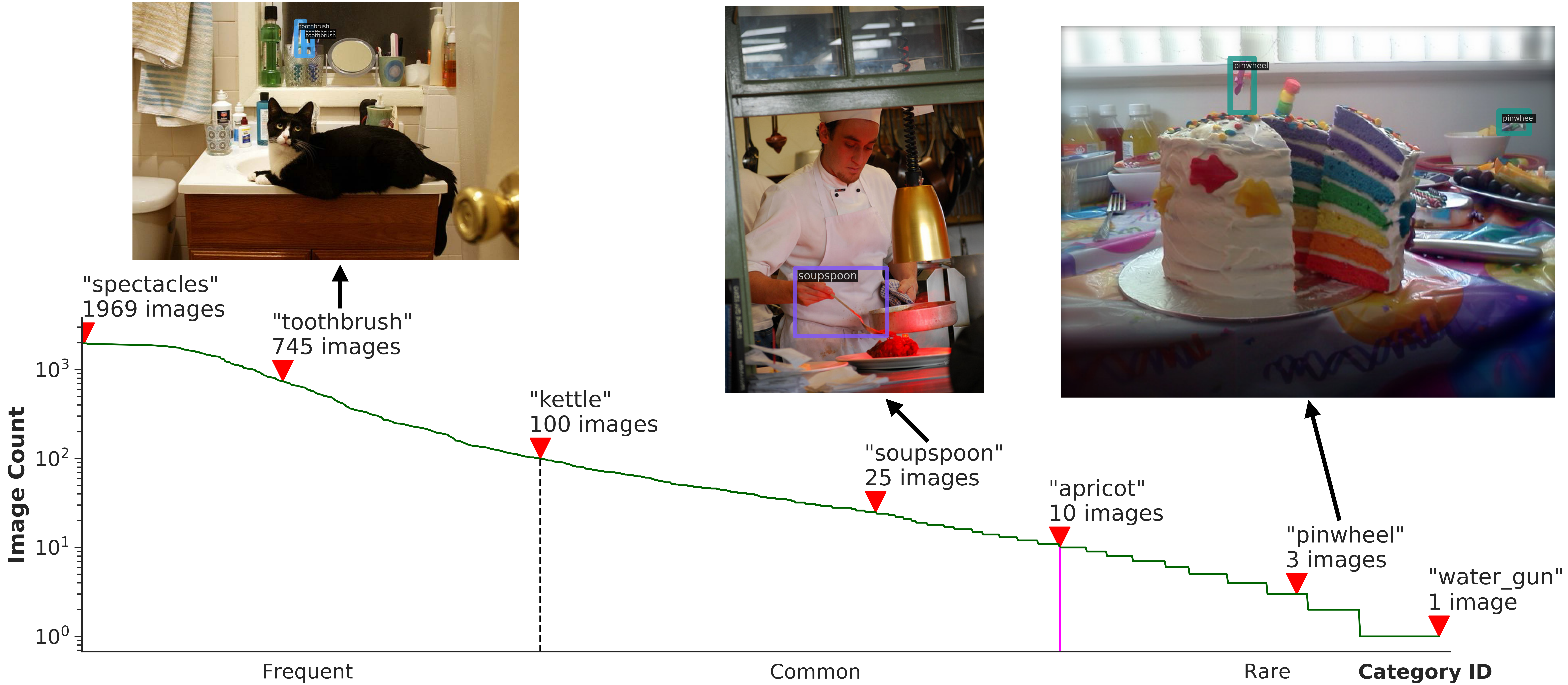
Outline

1. The Natural Long-Tailed Distribution
2. The Devil Is in the (De)Tail
3. Robust Performance Without Extra Labels

The Long-Tailed Distribution in Natural Scenes

LVIS: A Dataset for Large Vocabulary Instance Segmentation. Gupta et al., CVPR 2019.

The Long-Tailed Distribution in Natural Scenes



LVIS: A Dataset for Large Vocabulary Instance Segmentation. Gupta et al., CVPR 2019.

The Natural Long-Tailed Distribution

1. In a large and open vocabulary, many objects appear more often than others
2. More difficult than an *imbalanced dataset* because the "tail" distribution may consist of hundreds of rare objects having as few as a single exemplar
3. The long tail of rare objects follows the Zipf distribution and is ***inescapable***; collecting more images simply uncovers previously unseen rare categories

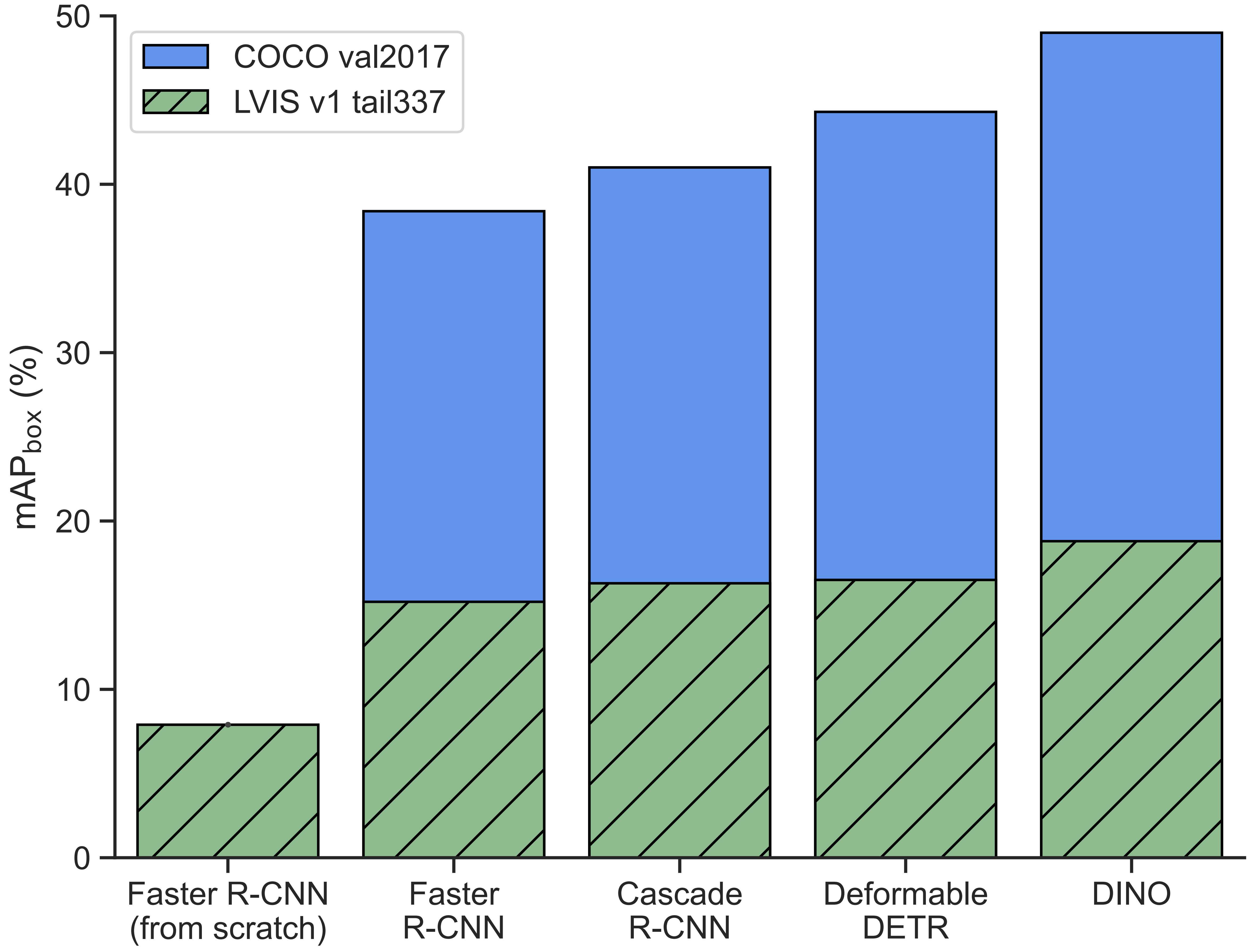
The Psycho-Biology of Language: An Introduction to Dynamic Philology. George Kingsley Zipf, Routledge 2013.

The Devil Is in the (De)Tail

How to Detect Genuinely Rare Objects?

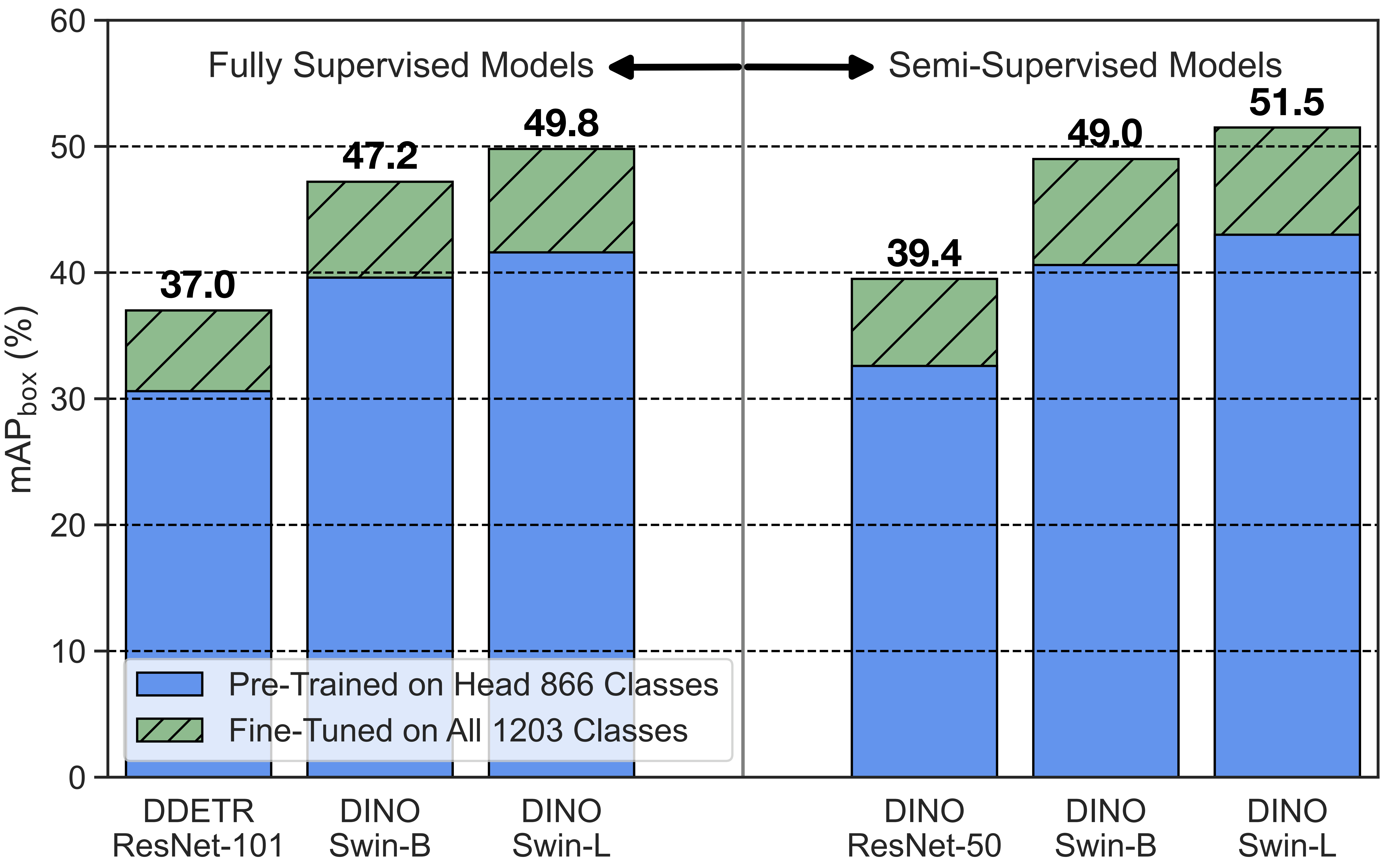
Intrinsically rare objects are strictly limited in observation and cannot be collected in more quantity from the open Internet.

Key Insight: Pre-Trained Representations Help



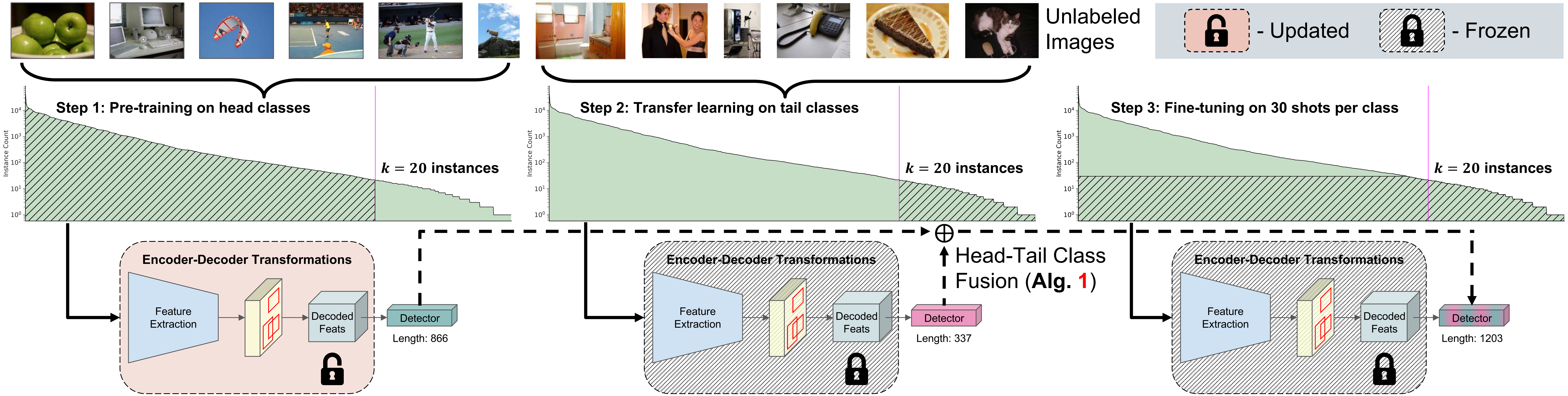
Transfer learning from pre-trained COCO representations (solid blue bars) helps improve rare class detection on LVIS (hatched green bars)

Key Insight: Pre-Trained Representations Help

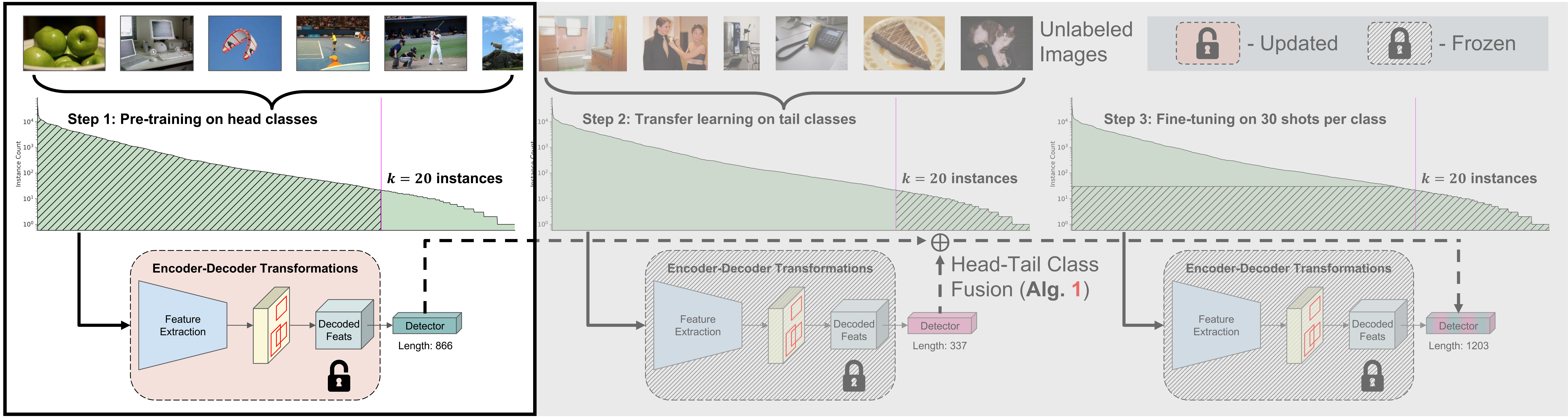


Our contribution: we find that stronger pre-trained head models (solid blue bars) are more effective long-tailed detectors (hatched green bars) across both supervised and semi-supervised settings

Our 3-Step Approach: Improved Head-to-Tail Model Transfer

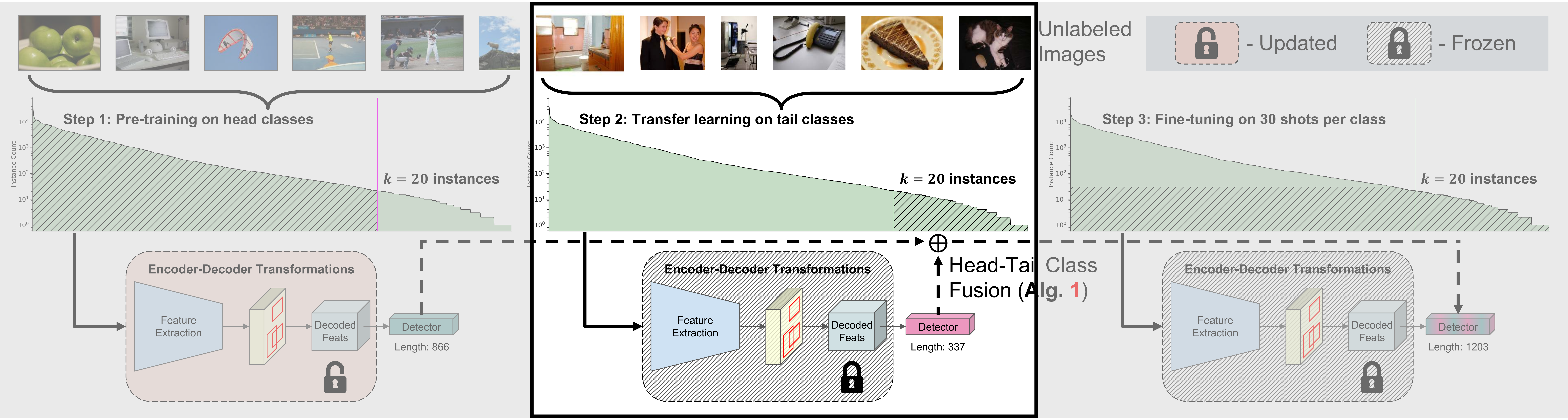


Step 1️⃣: Pre-Training on "Head" Classes



- Fully supervised pre-training with labeled examples
- Semi-supervised pre-training with both labeled and available unlabeled examples

Step 2️⃣: Transfer Learning on "Tail" Classes



- Instantiate the tail models by copying the head representations
- Freeze the pre-trained head representations except the detector
- Perform transfer learning on tail classes with optional unlabeled images

Step 2️⃣: Head-Tail Class Fusion

At this stage, we have two separate models with a shared representation

- One model optimized for accuracy on head classes
- The other optimized for accuracy on tail classes
- We wish to unify the two models into one for efficient single-pass inference on test samples containing both head and tail classes

Algorithm 1 PyTorch Pseudocode for Head-Tail Class Fusion.

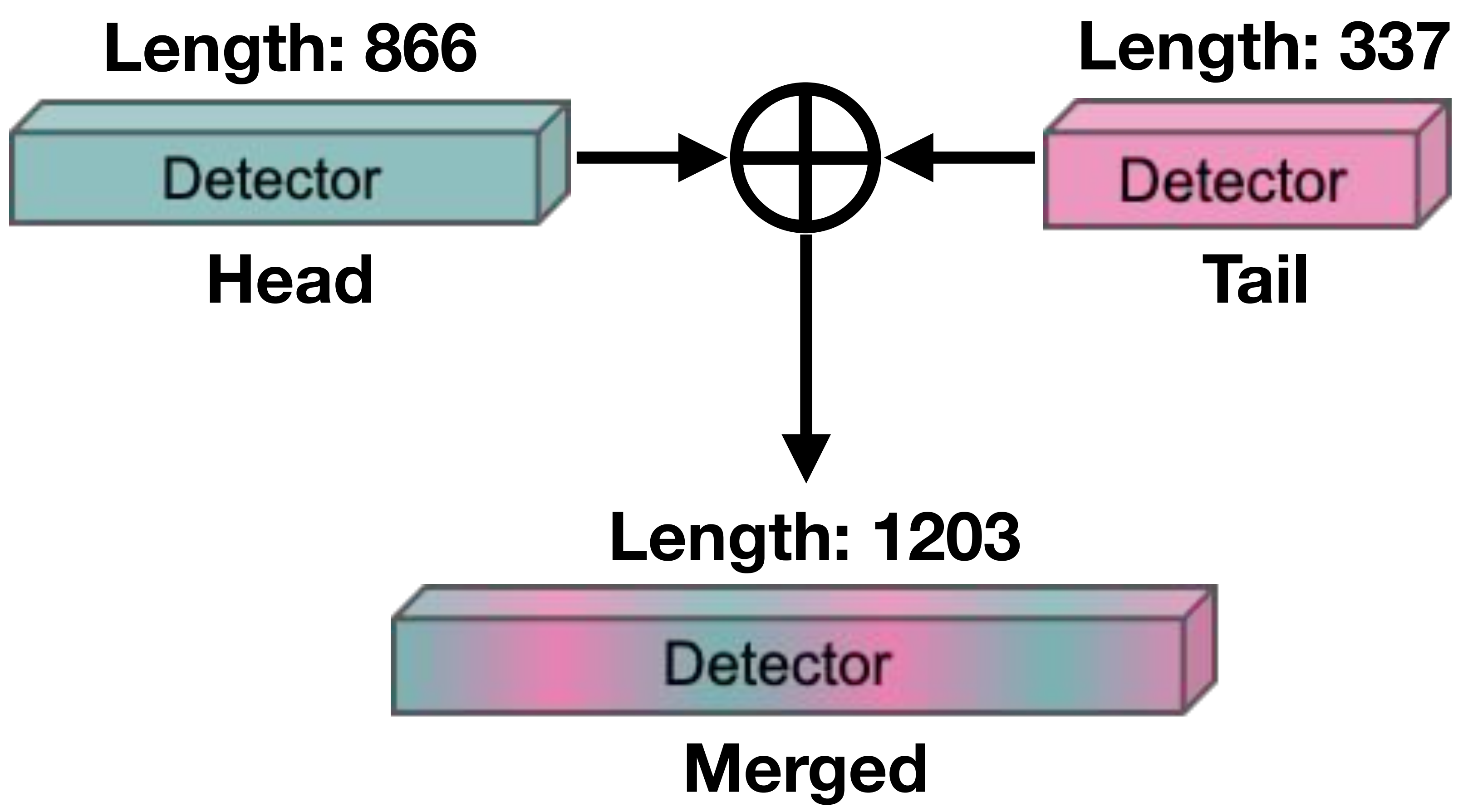
```
# HEAD_IDS : sorted list of head IDs, length 866
# TAIL_IDS : sorted list of tail IDs, length 337
# head_ckpt: model checkpoint on head classes
# tail_ckpt: model checkpoint on tail classes

ALL_IDS = sorted(HEAD_IDS + TAIL_IDS) # length 1203
ID2LABEL = {
   ID: label for label, ID in enumerate(ALL_IDS)
} # mapping from category ID to integer label
head_det = head_ckpt["state_dict"]["detector"]
tail_det = tail_ckpt["state_dict"]["detector"]
fused_det = torch.randn(len(ALL_IDS))

for label, ID in enumerate(HEAD_IDS):
   fused_det[ID2LABEL[ID]] = head_det[label]
for label, ID in enumerate(TAIL_IDS):
   fused_det[ID2LABEL[ID]] = tail_det[label]

head_ckpt["state_dict"]["detector"] = fused_det
torch.save(head_ckpt, save_filename) # to fine-tune
```

Step 2️⃣: Head-Tail Class Fusion



Algorithm 1 PyTorch Pseudocode for Head-Tail Class Fusion.

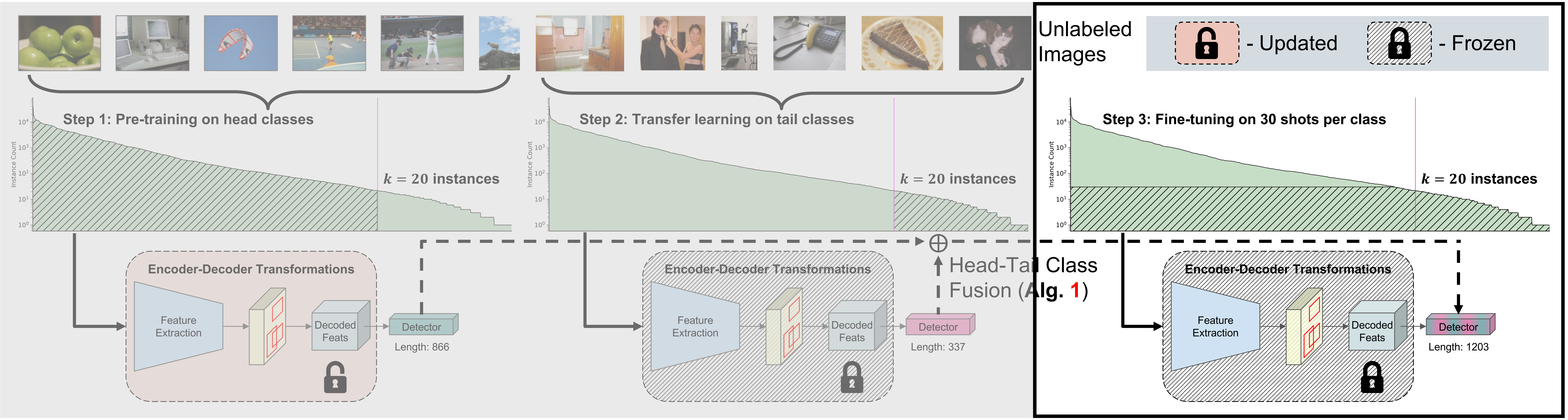
```
# HEAD_IDS : sorted list of head IDs, length 866
# TAIL_IDS : sorted list of tail IDs, length 337
# head_ckpt: model checkpoint on head classes
# tail_ckpt: model checkpoint on tail classes

ALL_IDS = sorted(HEAD_IDS + TAIL_IDS) # length 1203
ID2LABEL = {
   ID: label for label, ID in enumerate(ALL_IDS)
} # mapping from category ID to integer label
head_det = head_ckpt["state_dict"]["detector"]
tail_det = tail_ckpt["state_dict"]["detector"]
fused_det = torch.randn(len(ALL_IDS))

for label, ID in enumerate(HEAD_IDS):
   fused_det[ID2LABEL[ID]] = head_det[label]
for label, ID in enumerate(TAIL_IDS):
   fused_det[ID2LABEL[ID]] = tail_det[label]

head_ckpt["state_dict"]["detector"] = fused_det
torch.save(head_ckpt, save_filename) # to fine-tune
```

Step 3️⃣: Fine-Tuning on a Balanced Sample of Head and Tail Classes

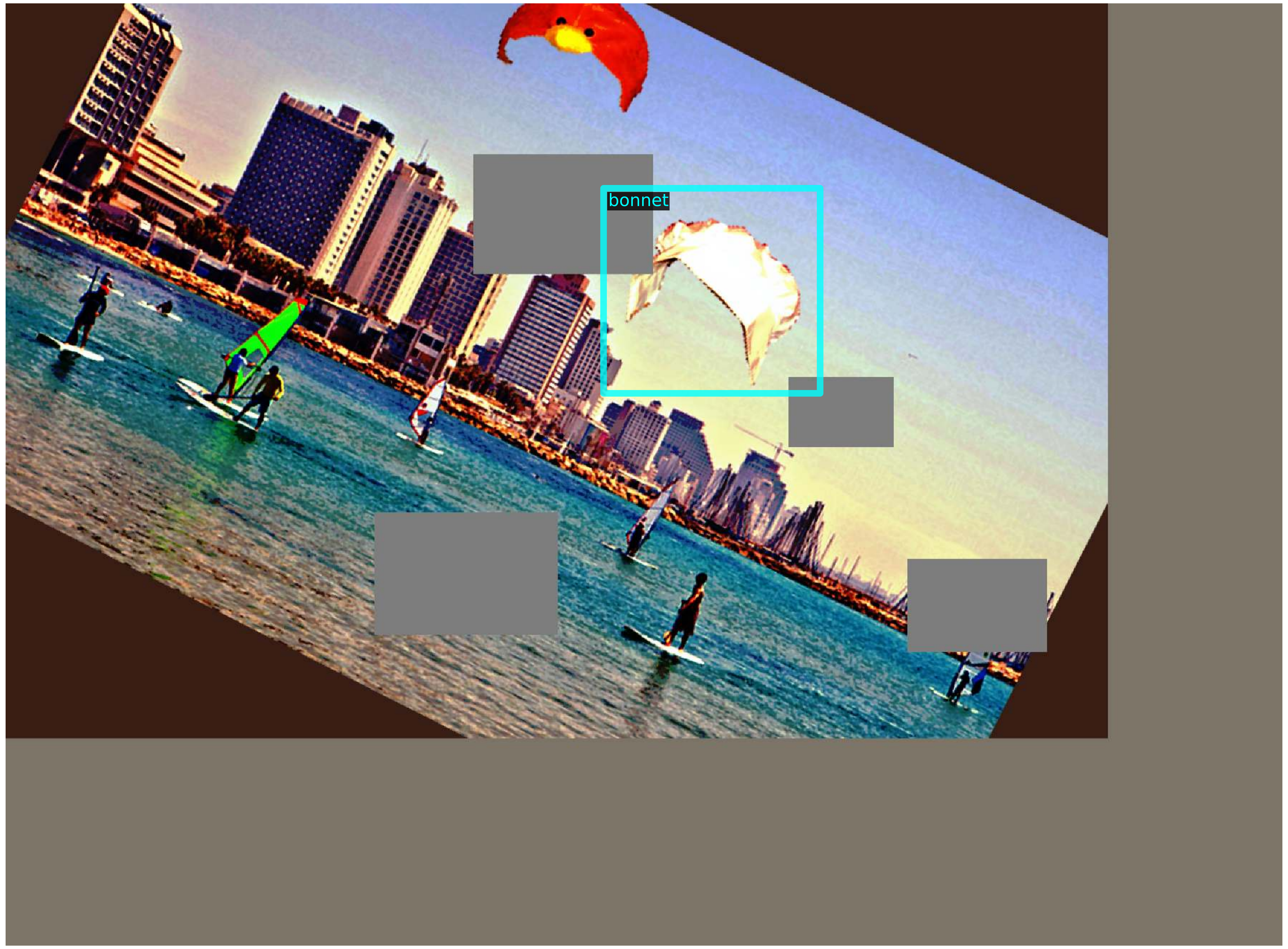


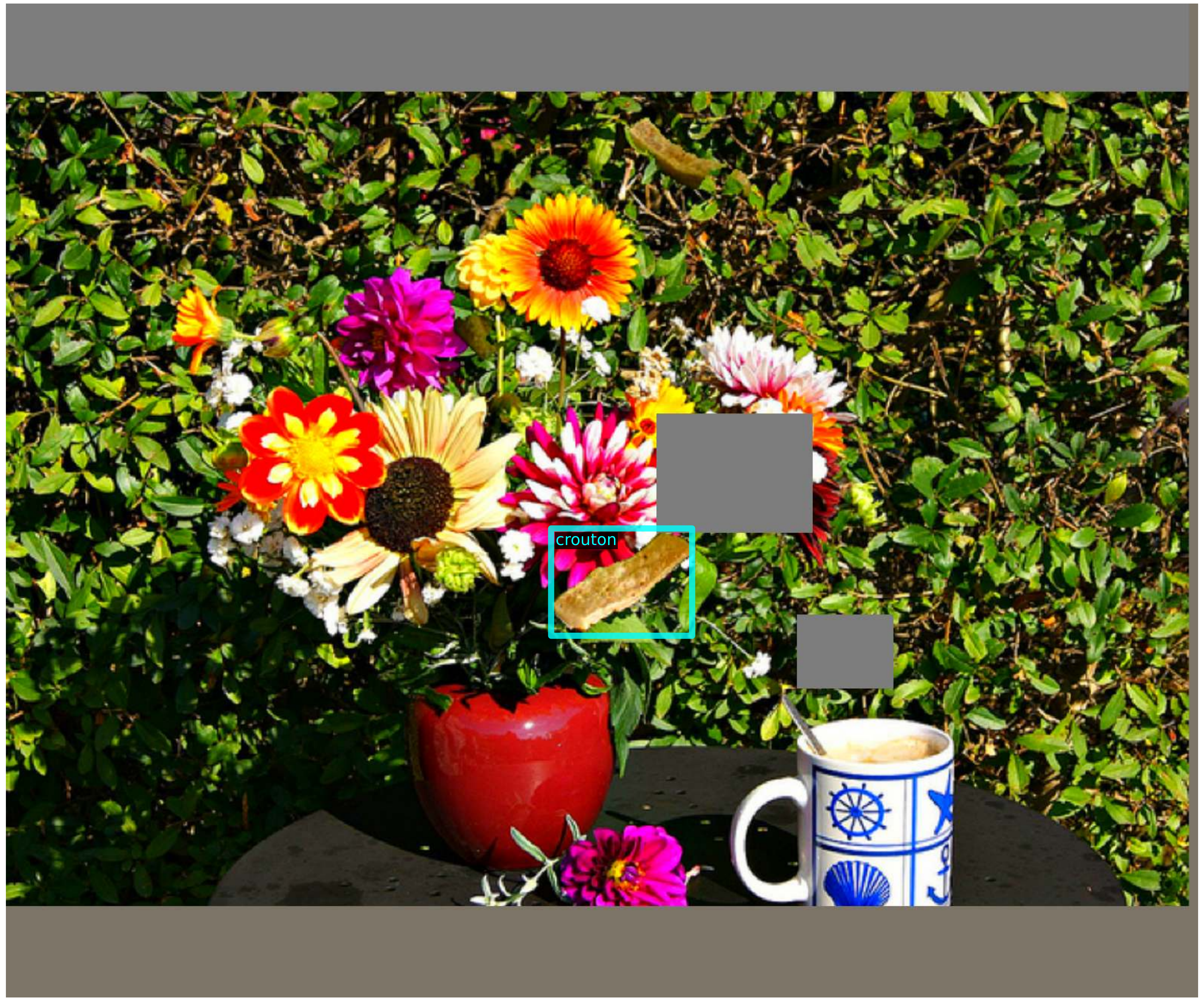
- Keep the pre-trained head representations frozen
- Fine-tune only the detector with a reduced learning rate to avoid catastrophic forgetting

What About the Intrinsically Rare Objects?

Compose complex ***fake scenes*** under strong photometric and geometric distortions
by pasting rare objects onto unlabeled images

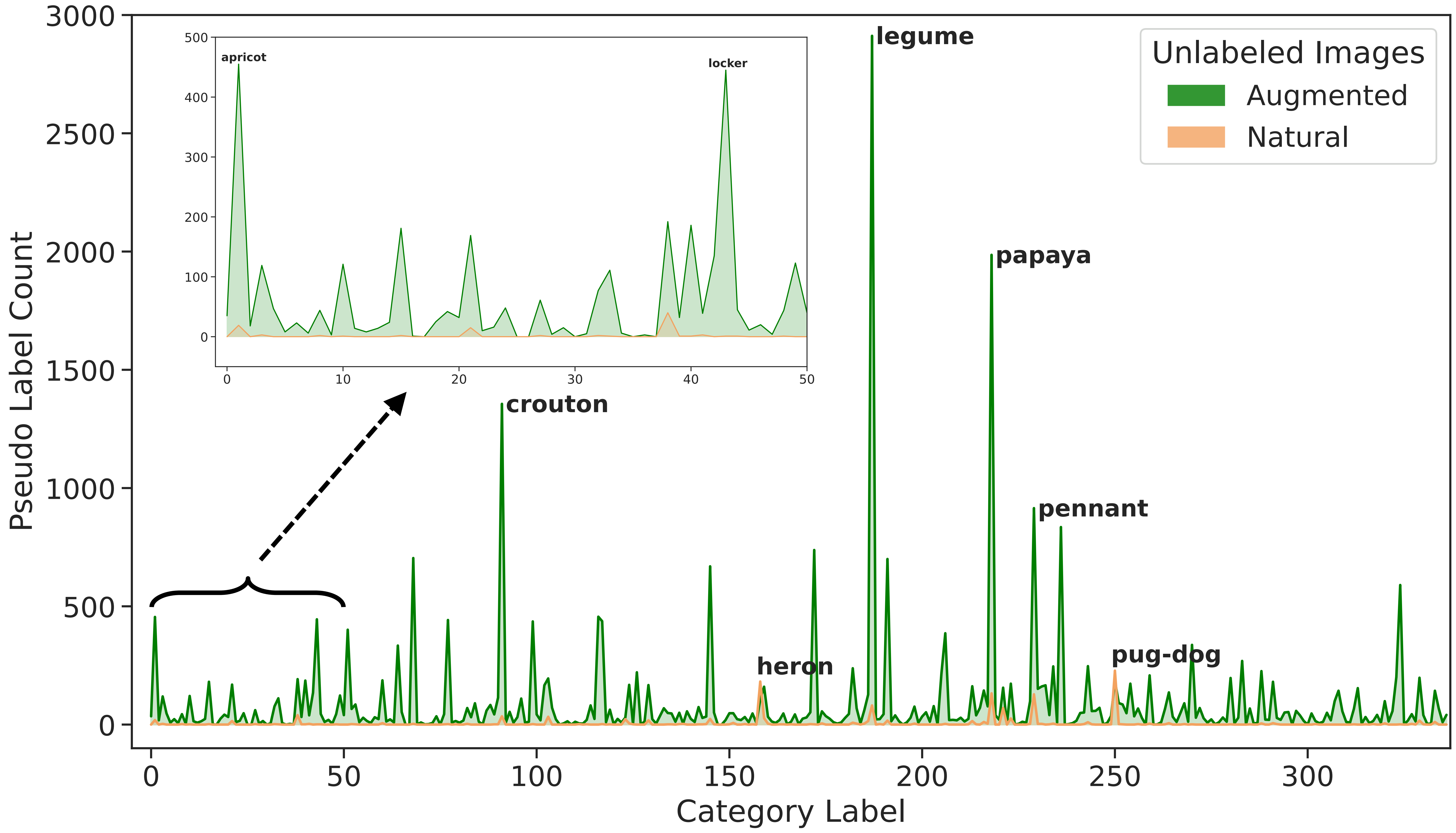
What About the Intrinsically Rare Objects?





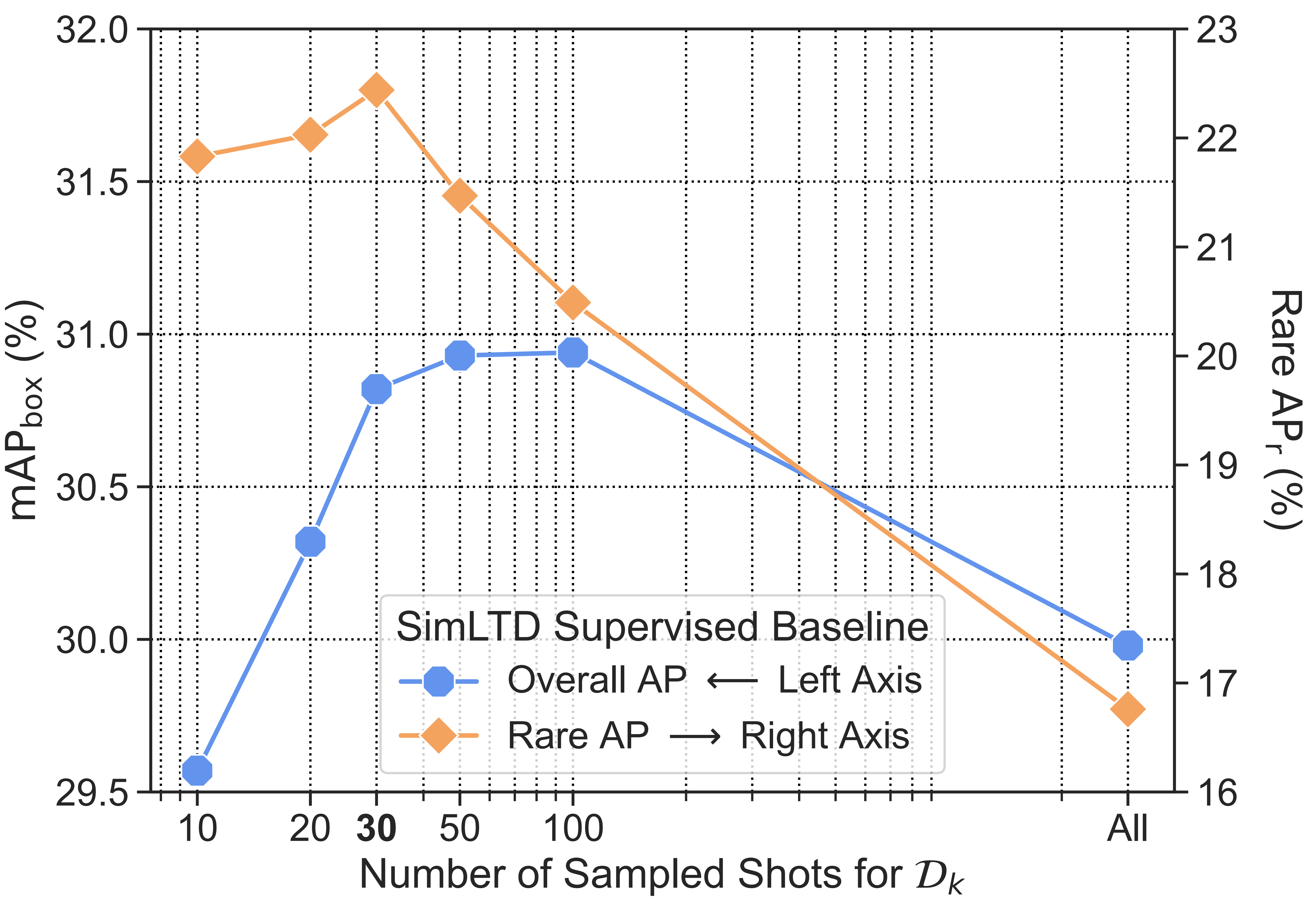
Compose complex ***fake scenes*** under strong photometric and geometric distortions
by pasting rare objects onto unlabeled images

What About the Intrinsically Rare Objects?



Augmenting unlabeled images with randomly pasted rare objects helps promote pseudo-labeling for effective semi-supervised learning

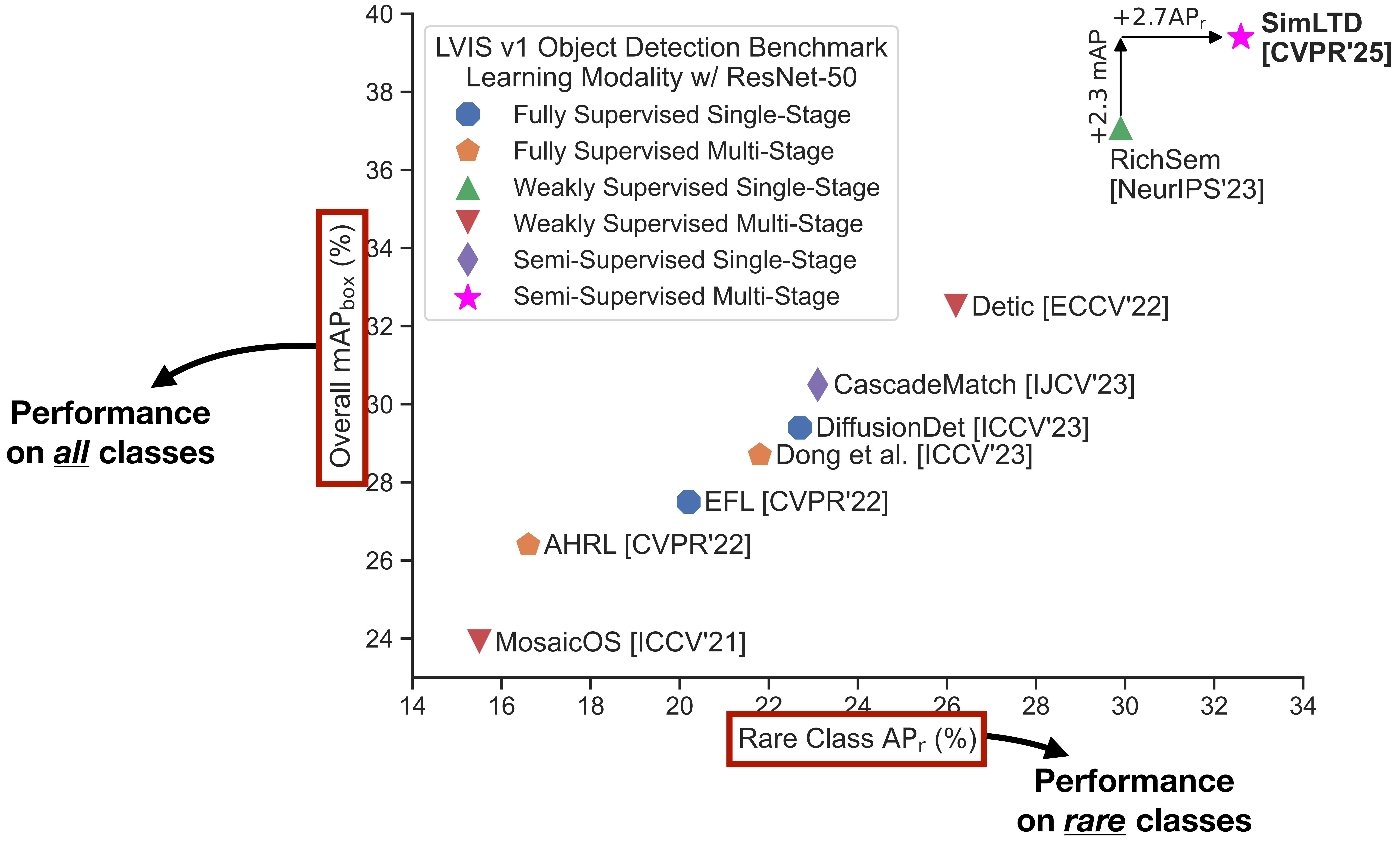
How Many Instances to Sample for Fine-Tuning?



30 instances, or shots, per class balance the trade-off between detection and catastrophic forgetting

🚀 Performance Evaluation 🔥

🚀 Advancing Long-Tailed Detection Without Extra Labels



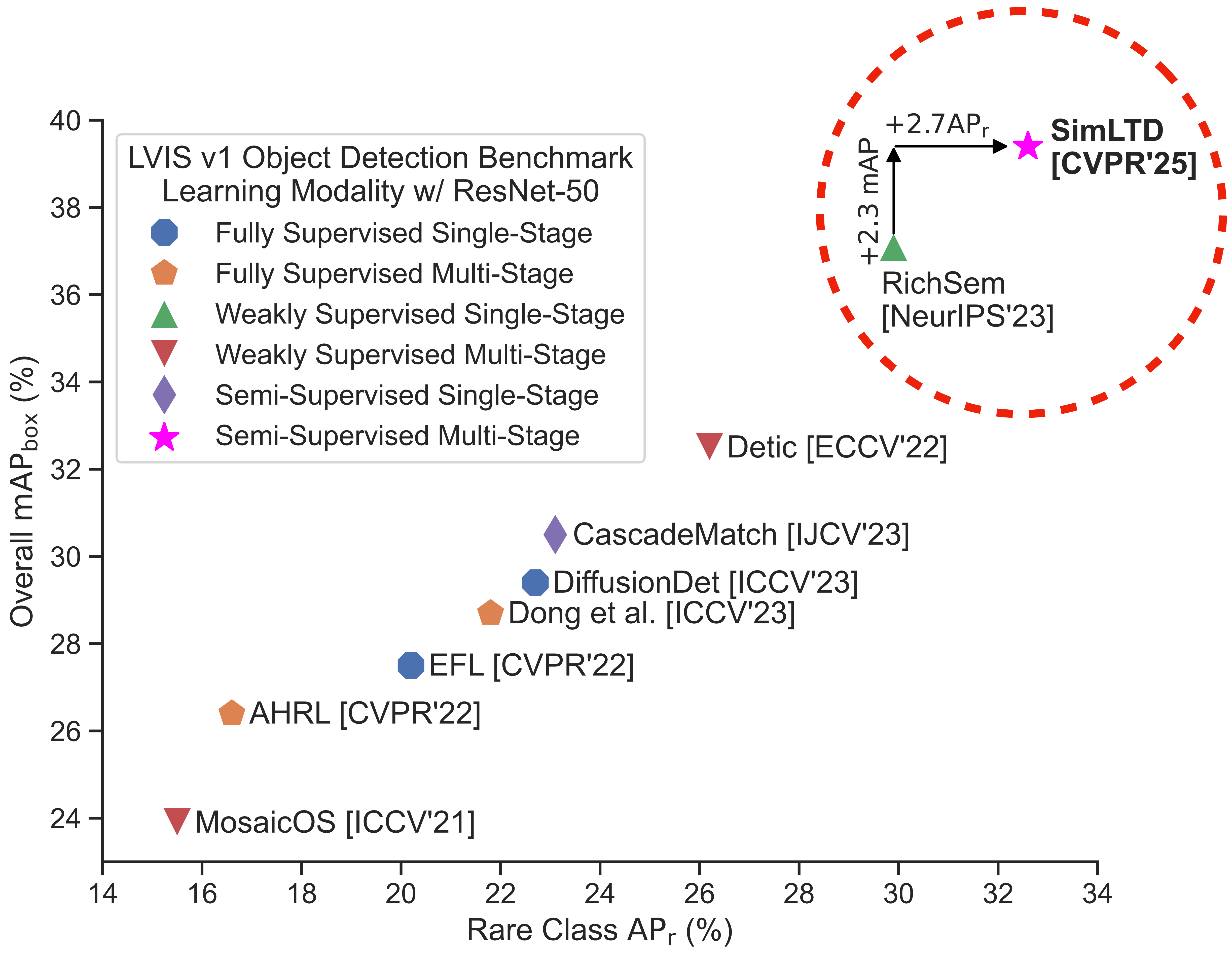
🚀 Advancing Long-Tailed Detection Without Extra Labels

❌ Fully supervised methods have limited performance without extra data

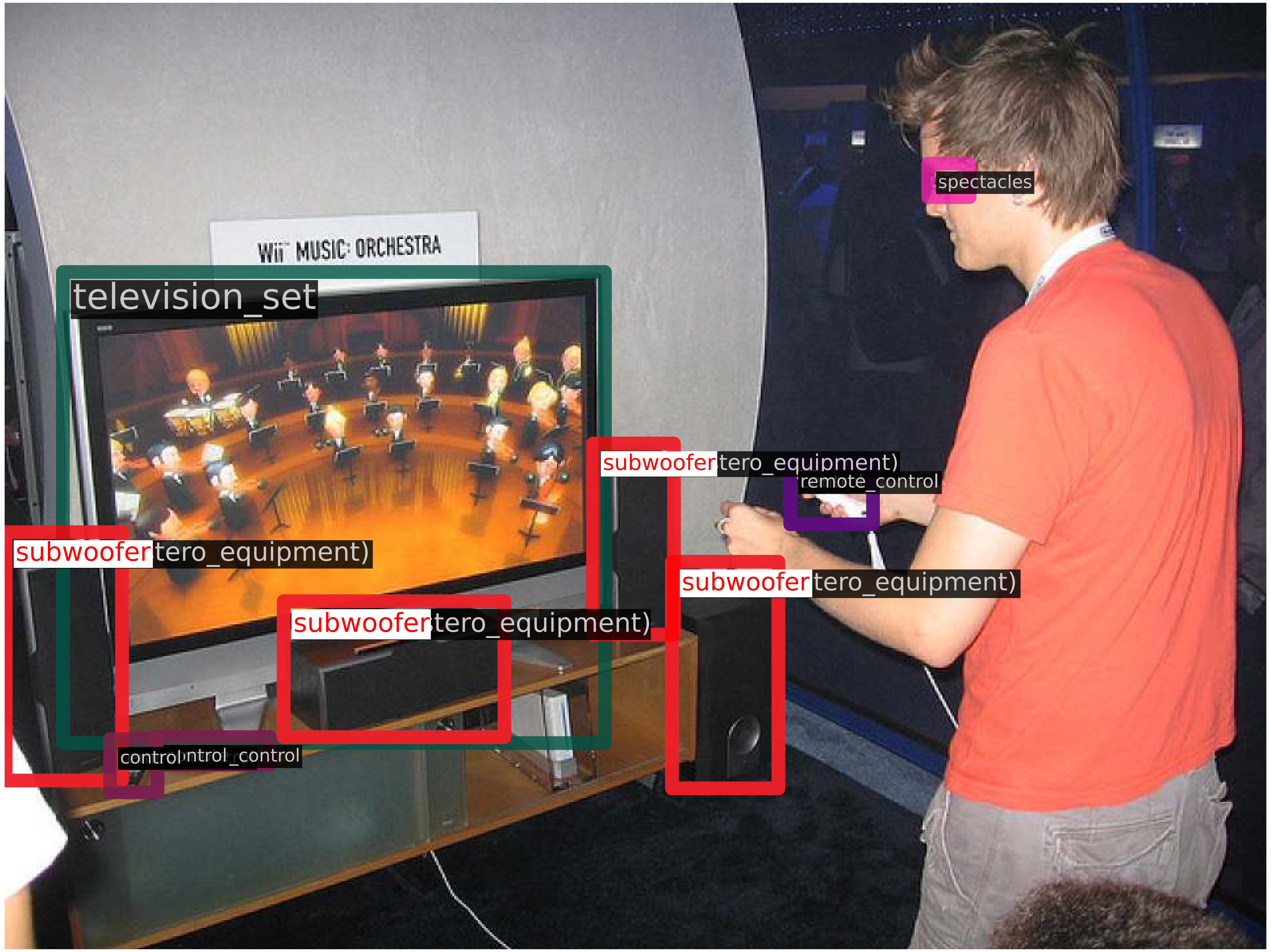
❌ Weakly supervised methods require extra data with whole-image labels

✅ Our semi-supervised method leverages extra unlabeled images in the wild

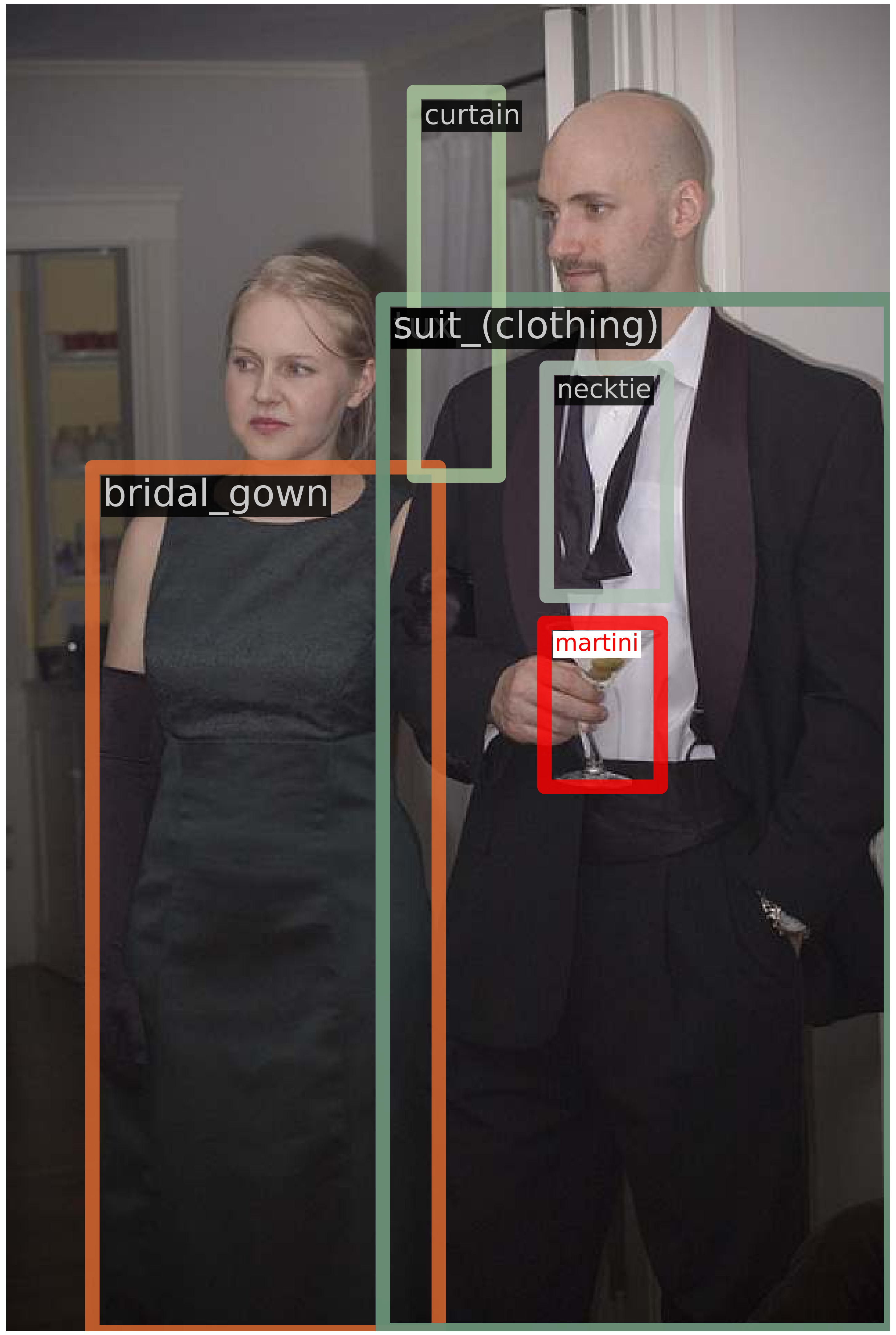
✅ ***<u>Benefit:</u> Unlabeled images are easy to collect without the burden of human annotations***

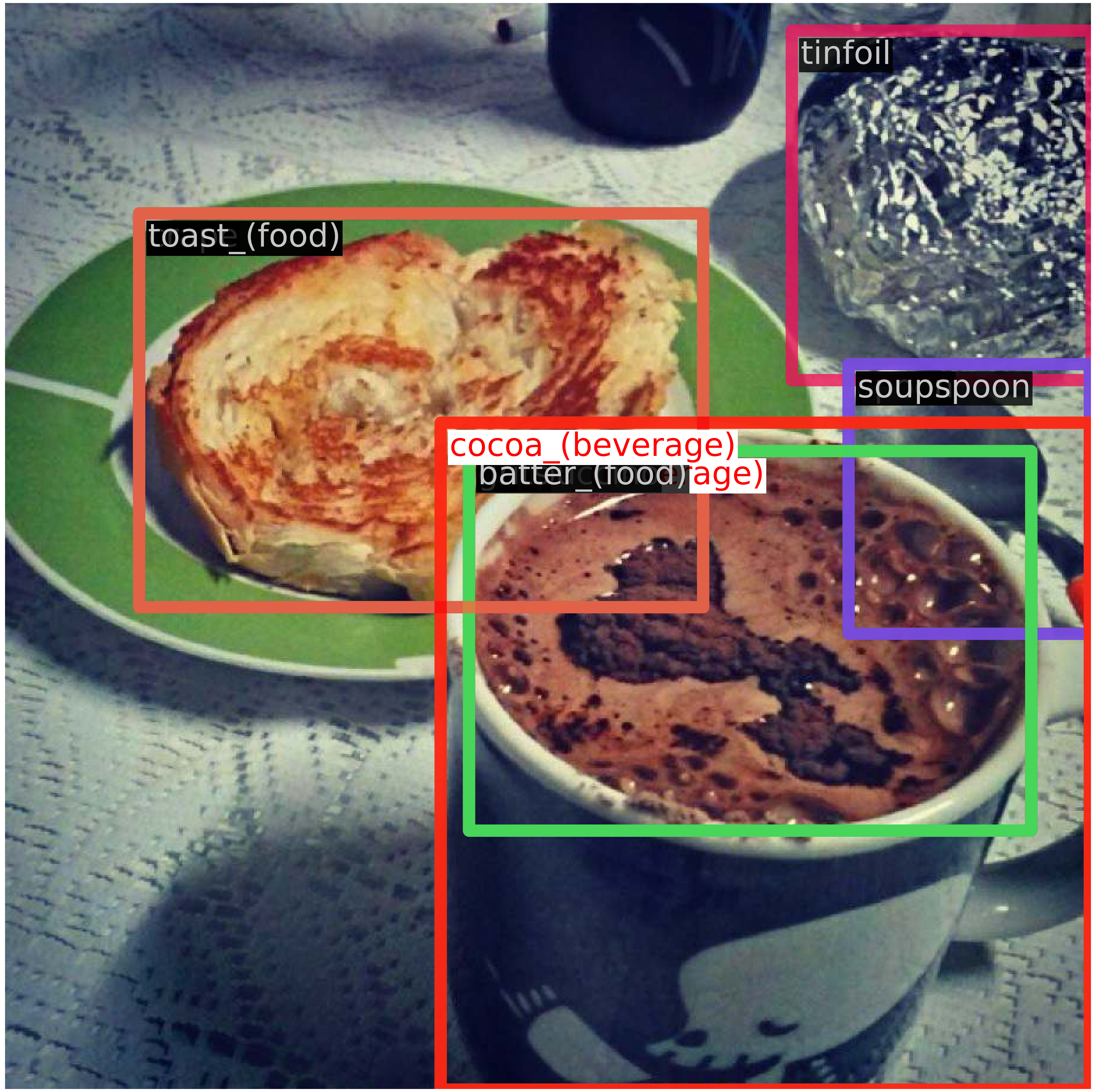


🔥 Robust Detection With a *Single* Training Exemplar (Red/White Box)

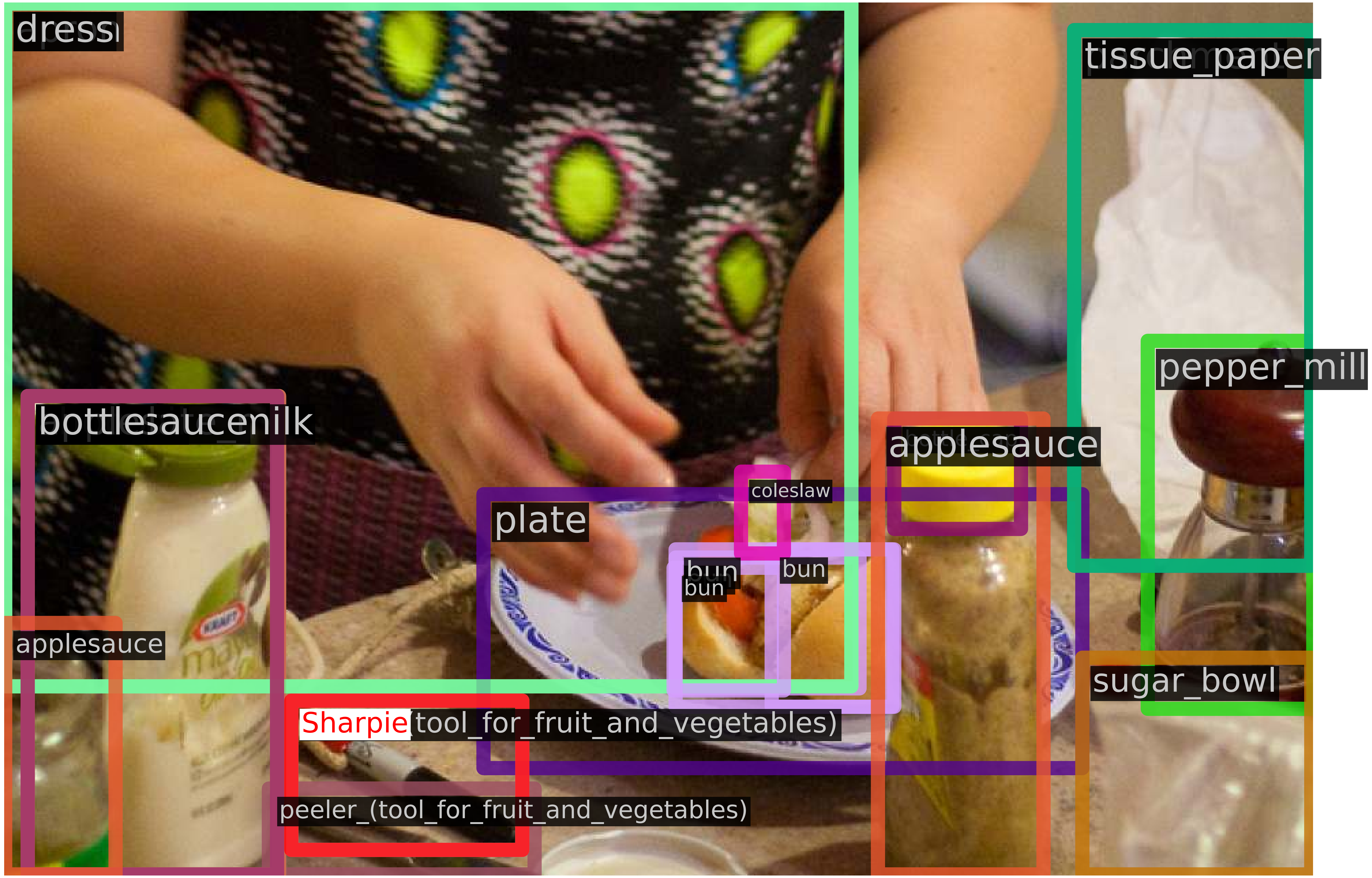


🔥 Robust Detection With *Three* Training Exemplars (Red/White Box)





🔥 Robust Detection With *Five* Training Exemplars (Red/White Box)



🔥 Robust Long-Tailed Detection Without Extra Image-Level Supervision