ObjectLab: Automated Diagnosis of Mislabeled Images in Object Detection Data

Ulyana Tkachenko [*1] **Aditya Thyagarajan** [*1] **Jonas Mueller** [1]

Abstract

Despite powering sensitive systems like autonomous vehicles, object detection remains fairly brittle in part due to annotation errors that plague most real-world training datasets. We propose ObjectLab, a straightforward algorithm to detect diverse errors in object detection labels, including: overlooked bounding boxes, badly located boxes, and incorrect class label assignments. ObjectLab utilizes *any* trained object detection model to score the label quality of each image, such that mislabeled images can be prioritized for label review/correction. Properly handling the erroneous data enables training a better version of the same object detection model, without any change in existing modeling code. Benchmarks on SYNTHIA and naturally-occurring annotation errors in COCO reveal that across different object detection models/datasets, ObjectLab consistently detects error with much better precision/recall compared to other label quality scores.

1. Introduction

Object Detection is a key computer vision task powering many high-impact applications where computers decide actions based on captured images via a learned model. The datasets used to train/evaluate these detectors require a massive amount of human labeling which is inevitably imperfect. Annotators of an object detection dataset inspect an image and, for each depicted object, are supposed to draw a bounding box around it and assign a discrete class label to categorize this object.

In real-world datasets, annotators make three types of mistakes depicted in Figure 1: (1) An **Overlooked** error in which a depicted object was not spotted and thus no corresponding bounding box around it exists in the given label for this image, (2) A **Badly Located** error in which annotators sloppily draw the bounding box around a depicted object such that its location/edges fail to perfectly enclose the object, (3) A **Swapped** error in which annotators draw a correct bounding box around a depicted object, but assign it the wrong class label. Such *Swapped* errors are also common in many classification datasets (Northcutt et al., 2021a), but the increased complexity of object detection annotation introduces potential for more varied types of label errors than encountered in classification. We propose an algorithm, **ObjectLab**, that utilizes *any* trained **object** detection model to estimate the incorrect **lab**els in such a dataset, regardless which of these 3 types of mistake the data annotators made.

Training and evaluating models with incorrect bounding box annotations is clearly worrisome. Likely mislabeled images in an object detection dataset should be reviewed and either re-labeled or excluded from the dataset. While some research advocates dealing with noisy labels by training models in a special manner (Nishi et al., 2021; Sukhbaatar & Fergus, 2014; Jiang et al., 2018; Zhang & Sabuncu, 2018), we advocate for a straightforward data-centric approach to improve the data directly by first estimating which images are mislabeled. The data-centric approach has many advantages over special modeling (Kuan & Mueller, 2022; Northcutt et al., 2021b) – most importantly, it can be used to improve the performance of *any* object detection model, regardless of its architecture or training strategy. Our ObjectLab approach[1] utilizes *any* trained detector to estimate *which* images are incorrectly labeled – these can be corrected to subsequently produce an even better version of this same model (without any change in the existing modeling code). This generality ensures data-centric methods like ObjectLab will remain a valuable asset in the computer vision toolkit long in the future, even when new architectures and training strategies have been invented which invalidate special modeling techniques developed specifically to accomodate today's models.

2. Methods

Our aim is detecting labeling issues in a standard object detection dataset, in which each image I is annotated with bounding boxes B around each depicted ob-

[*]Equal contribution [1]Cleanlab. Correspondence to: Jonas Mueller <jonas@cleanlab.ai>.

[1]Code to run our method: `https://github.com/cleanlab/cleanlab`
Code to reproduce benchmarks: `https://github.com/cleanlab/object-detection-benchmarks`

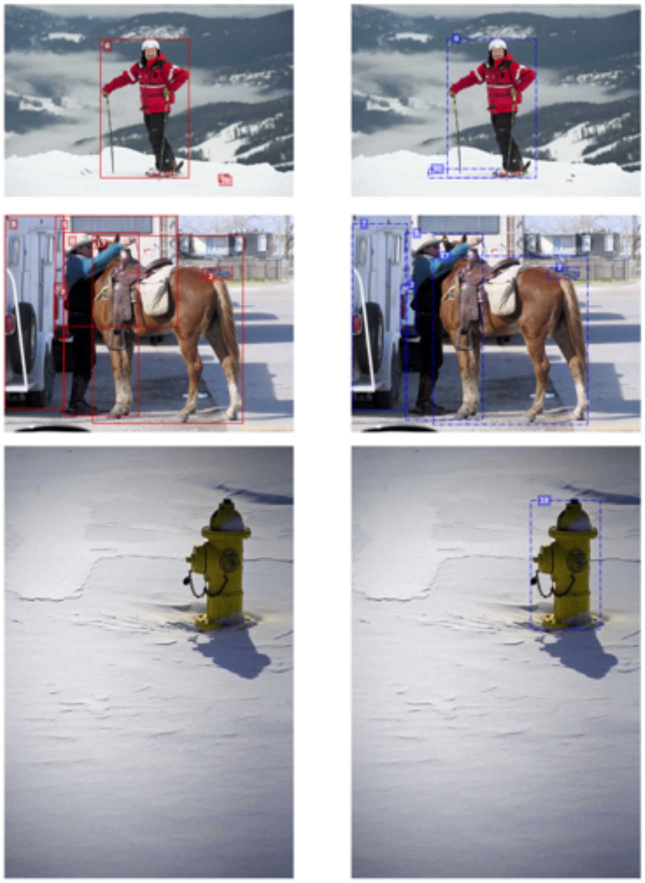

Figure 1. Images in COCO dataset with some of the *lowest* ObjectLab label quality scores. We show both the original given label (left column in red) and prediction from Detectron2-X101 model (right column in blue). These examples exhibit different naturally-occurring label errors: (top) *Overlooked* box, (middle) *Badly Located* box, (bottom) *Swapped* class label. In top row: annotators poorly outlined the **ski** (class #30) which the model localized much better (with confidence 0.835), leading to a low Badly-Located score in ObjectLab. In middle row: the vehicle on the left is incorrectly annotated as a **bus** (class #5), while the model predicted **truck** with confidence 0.996, leading to a low Swapped-score in ObjectLab. In bottom row: annotators missed the **fire hydrant** (class #10 in COCO) which the model detected with confidence 0.998, leading to a low Overlooked-score in ObjectLab.

ject, and each box is given an corresponding class label $c(B) \in \{1, \ldots, K\}$ categorizing the object into one of K classes. We interchangeably refer to the set of bounding boxes provided for image I in the original dataset, $\mathcal{L}(I)$, as the *given label* or *annotation* for this image.

Here we estimate a **label quality score** $\hat{s}(I)$ for each image, such that images receiving lower scores are more likely mislabeled (Kuan & Mueller, 2022), suffering from at least one of the aforementioned mistake-types (1)-(3). Practical constraints limit the number of images whose label can be reviewed, and thus an effective label quality score is key to prioritizing which images are worth another look. Good scoring methods ensure reviewers neither waste time inspecting correctly labeled images (high *precision*) nor fail to catch images with label errors (high *recall*).

A natural way to score label quality is via the predictions of a ML model. In this paper, we restrict ourselves to scoring methods that can be applied to the predictions from *any* standard object detection model, no matter its architecture or training strategy. A key question is: *when to trust the model over the original data*? To avoid bias from overfitting, label quality scores are computed using *out-of-sample* predictions from the model, i.e. based on predictions for an image I from a copy of the model which never saw I during training. We obtain out-of-sample predictions for every image in a dataset via straightforward 5-fold cross-validation.

Given an image I, a typical object detection model outputs *prediction* $\hat{\mathcal{P}}(I)$, which is a set of *predicted bounding boxes* $\hat{B}$, each localizing an estimated object and associated with: *predicted class* $\hat{c}(\hat{B}) \in \{1, \ldots, K\}$ and a *confidence* value $0 \leq \hat{p}(\hat{B}) \leq 1$ reflecting the estimated probability that the object localized by $\hat{B}$ belongs to class $\hat{c}(\hat{B})$. The set $\hat{\mathcal{P}}(I)$ is typically only comprised of predicted boxes $\hat{B}$ whose confidence exceeds some fixed threshold, i.e. $\hat{p}(\hat{B}) > \tau_{\downarrow}$. Here we use the same value $\tau_{\downarrow} = 0.5$ as the default in many popular object detection libraries, noting the empirical performance of ObjectLab was not significantly affected by trying smaller values of $\tau_{\downarrow}$ in our benchmarks. While certain types of object detection models output richer information than listed here (Zaidi et al., 2022), we believe being compatible with *any* type of model is key to an effective label quality score for data-centric AI. As object detection architectures advance over time and models become more accurate/calibrated, such label quality scores will remain applicable and immediately detect errors more effectively.

2.1. ObjectLab

Our proposed **ObjectLab** method straightforwardly scores each image I independently of the others and produces a label quality score $\hat{s}(I)$ solely based on the given label $\mathcal{L}(I)$ and model prediction $\hat{\mathcal{P}}(I)$. Algorithm 1 details our method. The ObjectLab score is a holistic representation of all possible labeling errors that can occur and is based on a geometric mean of three mistake-subtype scores $\hat{s}_{overlook}(I), \hat{s}_{badloc}(I), \hat{s}_{swap}(I)$ which respectively evaluate how likely this image suffers from an *Overlooked*, *Badly Located*, or *Swapped* error. These mistake-subtype scores are each computed by estimating a particular aspect of each annotated/predicted bounding box in image I via a particular quality score in [0,1], and subsequently pooling these quality estimates over all of the relevant boxes in I to form one of the subtype scores for I.

Algorithm 1 ObjectLab score $\hat{s}(I)$ for an image I

```
Require: given label $\mathcal{L}(I)$, model prediction $\hat{\mathcal{P}}(I)$
  $q_1, \ldots, q_N \leftarrow$ BadlocBoxScores($\mathcal{L}(I), \hat{\mathcal{P}}(I)$)
  $\hat{s}_{badloc}(I) \leftarrow$ softmin($q_1, \ldots, q_N$)
  $q_1, \ldots, q_N \leftarrow$ SwappedBoxScores($\mathcal{L}(I), \hat{\mathcal{P}}(I)$)
  $\hat{s}_{swap}(I) \leftarrow$ softmin($q_1, \ldots, q_N$)
  $q_1, \ldots, q_M \leftarrow$ OverlookedBoxScores($\mathcal{L}(I), \hat{\mathcal{P}}(I)$)
  $\hat{s}_{overlook}(I) \leftarrow$ softmin($q_1, \ldots, q_M$)
return Geometric mean of $\hat{s}_{badloc}, \hat{s}_{swap}, \hat{s}_{overlook}$ for $I$
```

To gain some intuition, consider say the *Badly located* mistake-subtype, for which we compute a particular quality estimate for each annotated bounding box B in I, reflecting the quality of its location. These location-quality estimates are then pooled over every annotated box B in I to form the single subtype score $\hat{s}_{badloc}(I)$, which roughly quantifies the estimated likelihood that any box B in I was badly located. Pooling via the mean quality estimate is overly sensitive to statistical variation the quality estimates for correctly located boxes, while pooling via the minimum quality estimate undesirably ignores the estimates for all boxes except one. An effective compromise between these extremes is *softmin* pooling (Wang & Mueller, 2022), in which we compute the pooled value $0 \leq \bar{q} \leq 1$ from a vector of per-box values $\vec{q} := \langle q_1, ..., q_N \rangle$ via the inner product: $\bar{q} = \langle \vec{q}, \text{softmax}(1 - \vec{q}) \rangle$. Here we suppose there are N annotated boxes for I. Such pooling reflects a softer version of the minimum function that still takes all boxes' quality estimates into account. Beyond the *Badly located* mistake-subtype, we also employ softmin pooling to aggregate certain per-box quality estimates into the other two types of subtype scores $\hat{s}_{overlook}(I), \hat{s}_{swap}(I)$.

Algorithms 2, 3, 4 detail the computation of the per-box quality estimates used for each subtype score. Intuitively, our quality score for an individual annotated box being badly located is based on its similarity with the nearest predicted box of the same class. Our quality score for an individual annotated box having a swapped class label is inversely related to its similarity with a nearby predicted box confidently predicted to belong to a different class. To estimate whether an individual predicted box $\hat{B}$ corresponds to an overlooked box that should have been in the original annotations, our corresponding quality score is based on this prediction's confidence and the similarity between $\hat{B}$ and the nearest annotated box of the same class as $\hat{c}(\hat{B})$.

Between any pair of bounding boxes in the same image, we define a **similarity** function:

$$sim(B_1, B_2) = \alpha \cdot k(B_1, B_2) + (1 - \alpha) \cdot IoU(B_1, B_2).$$

Here IoU is the standard *Intersection over Union* similarity measure and $k(\cdot, \cdot)$ is a Gaussian kernel similarity between 4D vectors defined by the outer edges of each box = $\exp(-||\vec{b}_1 - \vec{b}_2||/\sigma)$ where the entries of $\vec{b}_1$ (or $\vec{b}_2$) are the coordinates of the top-left and bottom-right corners of B_1 (or B_2) normalized to unit interval. Throughout we simply set $\alpha = \sigma = 0.1$ and did not find their precise values to affect performance. Rather the Gaussian kernel is included to avoid similarity ties when the IoU is equal to 0. Tied label quality scores between different images are undesirable as they do not aid in prioritizing which to review first. We also define sim_* as the **minimum possible similarity** between any pair of annotated and predicted boxes across all images in the dataset, and $q^* = 1$ as the **maximum possible quality estimate** for any box (across all mistake subtypes).

Computing Badly Located scores per annotated box. Badly located box scores are calculated for every annotated box via Algorithm 2. We simply score each annotated box based on its similarity with the nearest predicted box of the same class. When there is no such predicted box, we consider this annotated box well-located (assigning maximum quality score $q^* = 1$). For a well-trained model, we observed the majority of incorrect predictions are entirely false positive/negative detections; when objects are correctly detected, their predicted bounding boxes tend to be well-localized, except for classes where the original annotations also contain poorly located boxes thus confusing the model.

Computing Swapped scores per annotated box. Swapped box scores are calculated for every annotated box via Algorithm 3. We are most concerned that an annotated box may have a swapped class label when there exists an extremely similar predicted box that was predicted with high confidence to belong to a different class. For swapped errors, we score the annotated box quality based on the distance to the most similar predicted box that was confidently predicted to belong to a different class. If there are no such predicted boxes, we do not consider the annotated box to potentially have a swapped class label, and its quality estimate is set to the maximum value $q^* = 1$. Deciding what constitutes a highly confident prediction depends on a fixed threshold $\tau_\uparrow$, which one can set based on the estimated trustworthiness of the model (e.g. via a calibration curve). One can also use separate thresholds for each class (Northcutt et al., 2021b). Here we simply fix this threshold $\tau_\uparrow = 0.95$ corresponding to a 95% confidence value adopted as a de facto standard in statistical decision-making.

Computing Overlooked scores per predicted box. While the aforementioned quality estimates for the *Swapped* and *Badly Located* error types are computed for each annotated box, *Overlooked* errors are defined by the absence of such a box in the given label. Thus overlooked box quality scores are instead calculated for every predicted box via Algorithm 4. Again we only consider high-confidence pre-

Algorithm 2 BadlocBoxScores for image I

```
Require: given label $\mathcal{L}(I)$, model prediction $\hat{\mathcal{P}}(I)$
scores $\leftarrow \{\}$
for annotated box $B \in \mathcal{L}(I)$ do
    Let $k = c(B)$ denote its annotated class and $\mathcal{P}_k := \{\hat{B} \in \hat{P}(I) : \hat{c}(\hat{B}) = k\}$ denote the predicted boxes with the same predicted class.
    scores.append($q$)   where $q \leftarrow q^*$ if $\mathcal{P}_k = \emptyset$
                         else: $q \leftarrow \max_{\hat{B} \in \mathcal{P}_k} sim(B, \hat{B})$
return scores
```

Algorithm 3 SwappedBoxScores for image I

```
Require: given label $\mathcal{L}(I)$, model prediction $\hat{\mathcal{P}}(I)$
scores $\leftarrow \{\}$
for annotated box $B \in \mathcal{L}(I)$ do
    Let $k = c(B)$ denote its annotated class and $\mathcal{P}_{-k} := \{\hat{B} \in \hat{P}(I) : \; \hat{c}(\hat{B}) \neq k, \; \hat{p}(\hat{B}) > \tau_{\uparrow}\}$ be the predicted boxes with another predicted class whose confidence exceeds threshold $\tau_{\uparrow}$.
    scores.append($q$)   where $q \leftarrow q^*$ if $\mathcal{P}_{-k} = \emptyset$
                         else: $q \leftarrow 1 - \max_{\hat{B} \in \mathcal{P}_{-k}} sim(B, \hat{B})$
return scores
```

Algorithm 4 OverlookedBoxScores for image I

```
Require: given label $\mathcal{L}(I)$, model prediction $\hat{\mathcal{P}}(I)$
scores $\leftarrow \{\}$;
for predicted box $\hat{B} \in \hat{\mathcal{P}}(I)$ with $\hat{p}(\hat{B}) > \tau_{\uparrow}$ do
    Let $k = \hat{c}(\hat{B})$ denote its predicted class and $\mathcal{L}_k := \{B \in \mathcal{L}(I) : c(B) = k\}$ denote the subset of annotated boxes for class $k$.
    if $\mathcal{L}_k = \emptyset$ then
        $q = sim_* \cdot \big(1 - \hat{p}(\hat{B})\big)$
    else
        $q = \max_{B \in \mathcal{L}_k} sim(B, \hat{B})$
    scores.append($q$)
if scores $= \emptyset$:   scores $= \{q^*\}$
return scores
```

dicted boxes whose confidence exceeds threshold $\tau_{\uparrow}$. For each such box, we consider whether there is a corresponding annotated box present in the image or not. When there is, the similarity between the two serves as the quality score, otherwise we use the minimum possible similarity adjusted by the model confidence (since a 99% confident prediction with no corresponding annotated box is more indicative of an overlooked error than a 98% confidenct prediction).

Aggregating separately quantified evidence for these various types of potential errors, the ObjectLab score offers a model-agnostic and computationally-efficient estimate of label quality in object detection datasets. This method straightforwardly utilizes predictions from a trained model. Sorting images by our proposed score can help detect a wide variety of labeling errors of different types.

3. Related Work

Several previous works have demonstrated object detection datasets are full of labeling errors, mostly via manual analysis (Murrugarra-Llerena et al., 2022; Ma et al., 2022; sama, 2022; Hasty.ai, 2022). Other work has focused only on specific error types in object detection data and model-specific techniques to improve training with noisily labeled data (Xu et al., 2019). Due to the immense value of systematic label error detection, extensive research has developed methods for this task particularly for classification datasets (Brodley & Friedl, 1999; Muller & Markert, 2019). *Confident Learning* (Northcutt et al., 2021b) is one particularly popular methodology to automatically detect mislabeled classification data. Recent work has studied methods to extend these label error detection capabilities beyond classification to structured data in NLP (Klie et al., 2022) and segmentation data in computer vision (Rottmann & Reese, 2022; Chan et al., 2021). In these areas, label quality scores have been found to be effective (Kuan & Mueller, 2022), particularly when they are properly pooled for data with complex multi-dimensional labels, e.g. via the softmin pooling used in ObjectLab (Wang & Mueller, 2022; Thyagarajan et al., 2023).

Some of the related label quality scoring methods discussed in this section are also considered as baseline methods for comparison in our subsequent benchmarks. We restrict our attention to general approaches like ObjectLab that can be used with any standard model and training strategy, because the score is produced solely based on model predictions and the original labels. Throughout all such methods are only applied to out-of-sample predictions obtained via cross-validation.

3.1. mAP label quality score

In *error analysis*, one sorts the data by the predictions' loss according to a standard evaluation metric computed separately for each instance. The resulting ranking reveals images for which the model struggles most, often because some of them are mislabeled (Bolya et al., 2020; Voxel51, 2023; Klie et al., 2022). Thus this constitutes a reasonable approach to label quality scoring for general ML tasks.

Mean Average Precision (mAP) is a widely used evaluation metric in object detection, quantifying the accuracy of an object detector via both its precision (percentage of correctly identified objects out of all the predicted objects) and recall (percentage of correctly detected objects out of all the annotated objects). mAP is computed by calculating the Average Precision (AP) for each class and then taking the mean across all classes. As a standard approach to error analysis in object detection, we can compute mAP per image, and also use this as an alternative label quality score (Voxel51, 2023). If one had a perfect model, label quality scoring via mAP would appropriately measure *how wrong* the data annotations are. But in practice, model predictions will never be perfect, especially from models trained with a noisily labeled dataset.

3.2. Tile-estimates approach

Beyond error analysis with per-image mAP scores, we also consider a direct extension of methods for identifying label errors in classification tasks (Northcutt et al., 2021b; Kuan & Mueller, 2022) to the object detection setting. To do so, we simply reduce aspects of object detection to a classification perspective. A straightforward way is a tile-based reduction, in which we first divide each image into a grid of size (J, J). Each tile in this grid is assigned a label and predicted class probabilities, and across all the images, these tiles are treated as separate instances in a classification task (ignoring which image each tile stems from). Subsequently standard label-quality scoring for classification (Kuan & Mueller, 2022) can be applied to score each tile. Here we simply use the likelihood of the given label according to the predicted class probabilities. To get a label quality score for an image, we finally pool the tile-scores over this image. We explored various pooling options and found that a simple geometric mean to work well.

To assign labels for each tile based on the given object detection annotations, we compute the overlap between tiles and annotated bounding boxes and assign the original bounding box label to the tiles significantly overlapping with this box. To obtain predicted class probabilities for a tile from our object detection model outputs, we form a kernel-smoothing predictor within each image. Here we apply the aforementioned similarity function $sim()$ between boxes but here to a tile and each bounding box, in order to construct a similarity-weighted average of all boxes probabilities (for this same image) as the predicted class probability vector for the tile.

3.3. CLOD (Chachuła et al., 2022)

Similar to our above extension of classification label quality scores to the object setting, Chachuła et al. (2022) propose an extension of the *Confident Learning* (Northcutt et al., 2021b) approach to detect label errors in object detection. Their CLOD method involves clustering annotated and model-predicted boxes based on IoU distance – specifically single linkage agglomerative clustering. As in our tile-estimates approach, CLOD assigns a label and predicted probabilities to each cluster, which allows the application of Confident Learning to assess label quality (treating each cluster as a separate instance). Subsequently, one can simply use mean-pooling over the clusters within an image to obtain a label quality score for the image. Chachuła et al. (2022) also consider the Overlooked, Swapped, and Badly Located object detection errors and propose CLOD as an effective way to detect them.

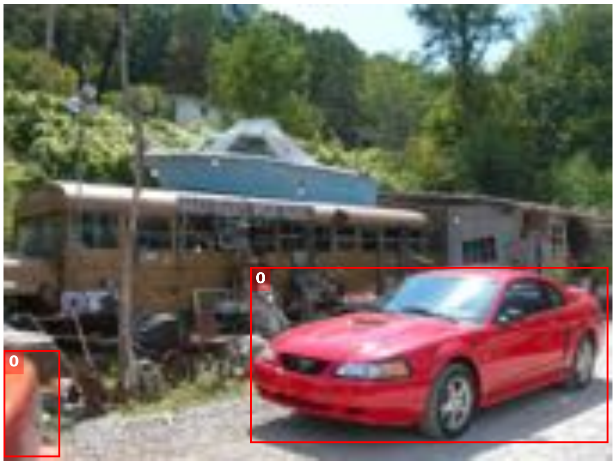

Figure 2. Example of a naturally mislabeled image in the COCO-bench dataset that receives low ObjectLab score. We show the original dataset label, which contains no boxes beyond those depicted for **Car** (class #0). Here the seated **person** and the **chair** were overlooked in the given label, even though these are among the 5 COCO-bench classes.

4. Experiments

4.1. Dataset and Models

Our benchmarks evaluate label quality scoring methods by training two object detection models across three datasets in order to ensure the results are dataset and model agnostic. Table 2 reports the accuracy of each model on each dataset. Our first two datasets (COCO-bench and SYNTHIA) are specially curated for evaluation by ensuring we know which images are truly mislabeled or not.

COCO-bench dataset. This subset of 2171 images from the famous COCO 2017 dataset (Lin et al., 2014) only considers 5 of the classes: person, chair, cup, car, and traffic light. These images and classes were selected because two other groups have re-annotated these images considering these classes (Ma et al., 2022; sama, 2022), and we use these redundant annotations here to determine which of these images truly have an annotation error in COCO. Specifically for each image, we compared its 5-class COCO annotation

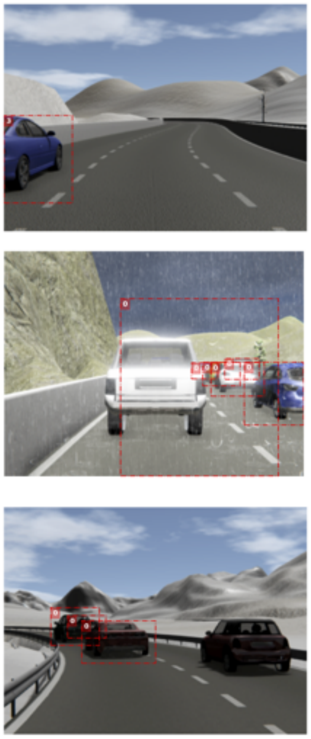

Figure 3. Various errors in our SYNTHIA-AL dataset, including: class label for **Car** (class #0 in SYNTHIA-AL) swapped with **Bicycle** (class #3), bounding box around depicted **Car** shifted to incorrect location (middle), and omitted bounding box around depicted **Car** (bottom). These examples involve the **Car** class, but similar errors exist in labels for each of the other 4 classes.

to its independent annotation by Ma et al. (2022) and by sama (2022) in order to determine ground truth. When both of these extra annotations disagreed with the COCO annotation but agreed with one another, we considered the image to be mislabeled in our COCO-bench dataset (whose labels all come from 5-class COCO, not the extra annotations). When all three annotations agreed, we considered the image correctly labeled. Here *agreement* between annotations was assessed by thresholding their pairwise mAP score. We manually inspected the remaining images to decide which were mislabeled or not. In total, 251 images in COCO-bench are considered mislabeled and we are confident in the ground truth assessments. Figure 2 depicts an example from this benchmark that this process revealed to be truly mislabeled.

SYNTHIA-AL dataset. Bengar et al. (2019) curated the SYNTHIA-AL dataset as an object detection benchmark where the ground truth labels are known because the images are synthetically generated by a realistic graphics engine (Ros et al., 2016). To ensure more independent images from what was originally a video dataset, we ensured a minimum distance of 28 frames between any pair of images included in our benchmark dataset. Our benchmark version of this dataset contains 5000 images and 5 classes: Pedestrian, TrafficLight, Car, TrafficSign, Bicycle.

We then randomly perturbed some of the clean labels in this dataset to inject various types of mislabeling: dropped bounding boxes, swapped class labels, and shifted bounding boxes. Some images contained more than one type of annotation error with 22% unique images containing at least one error. Our perturbations were considered as sole source of ground truth error for the SYNTHIA-AL dataset, given the images are generated by a graphics engine. Figure 3 depicts some examples from this benchmark. While we cannot characterize all properties of the naturally-occurring label errors present in COCO-bench, we control the label errors in SYNTHIA-AL, facilitating more systematic evaluation.

COCO-full dataset. Finally we also considered detecting mislabeled images in the full COCO 2017 training dataset, which has 80 classes (Lin et al., 2014). We refer to this dataset of 118,000 images as *COCO-full* to distinguish it from COCO-bench. Figure 1 shows some of the label errors automatically detected in this dataset. Note that we do know the ground truth mislabeled images in COCO, and thus only performed a limited evaluation with this dataset. For each label quality scoring method, we manually reviewed its 100 lowest-scoring images in COCO to assess what fraction of them were actually mislabeled.

Detectron-X101 model. The Detectron-X101 network is one of the most accurate object detection models in the popular Detectron2 library (Wu et al., 2019). This model uses a ResNeXt backbone (Xie et al., 2017) with a Feature Pyramid Network (Lin et al., 2017); standard convolutional and fully-connected output heads are used for box prediction.

Faster-RCNN model. Proposed by Ren et al. (2015), the Faster R-CNN architecture is one of the most widely used object detection methods. Here we specifically use the R-50-FPN Faster-RCNN network from the MMDetection library (Chen et al., 2019). This model shares parameters between a fully-convolutional region proposal network and the detection network, which is based on a ResNet-50 (He et al., 2015) backbone with a Feature Pyramid Network (Lin et al., 2017). Faster-RCNN is slightly less accurate than Detectron-X101 (Table 2) but often favored for its efficiency.

4.2. Evaluation Metrics

Our primary interest is how well our label quality estimates correctly prioritize images that have annotation errors over those which do not. Label error detection can be viewed as a form of information retrieval, a field with standard evaluation metrics based around precision/recall, which we

Table 1. Precision@100 achieved by various label quality scores for detecting mislabeled images in the COCO-full dataset.

MODEL	OBJECTLAB	MAP
DETECTRON-X101	0.58	0.22
FASTER-RCNN	0.42	0.17

adopt here to evaluate different label quality scoring methods (Kuan & Mueller, 2022). For each set of label quality scores, we compare them against the ground truth information about which images are mislabeled to compute their *Average Precision*, *Precision @ 100* (i.e. what fraction of the 100 lowest-scoring images are truly mislabeled), and *Precision @ T*, for $T =$ the true number of mislabeled images in each dataset. For the full COCO dataset where this ground truth information is not available, we only report *Precision @ 100* for a select number of label quality scoring methods (as it is labor intensive to report).

5. Results

Figure 5 shows that ObjectLab detects mislabeled images with better precision/recall than other label quality scores in both COCO-bench and SYNTHIA-AL, regardless of which object detection model is used. Out of the other label quality scores evaluated, the basic mAP score performs the best on SYNTHIA-AL but does not fare as well on COCO-bench.

Table 1 reports the results of our label quality score evaluation in the full 80-class COCO dataset. Because we could only calculate the precision via laborious manual review of top-ranking images under each method, we limit this evaluation to the *Precision @ 100* metric and compare ObjectLab against the straightforward mAP label quality score. In COCO-full, ObjectLab again consistently detects label errors with much higher precision than the mAP score across both types of models.

While ObjectLab explicitly accounts for model confidence and specific forms of errors expected in practice, the mAP score exclusively relies on the IoU between predictions and labels. Figure 4 shows examples where mAP label quality scores mistakenly flag correctly labeled images due to imperfect predictions from the object detection model. Unlike mAP, ObjectLab scores are unaffected by bad model predictions made with insufficient confidence. Unlike mAP, ObjectLab scores do not explicitly penalize images with annotations for which there is no corresponding prediction. We did not encounter images with extraneously added bounding boxes in our examination of object detection datasets.

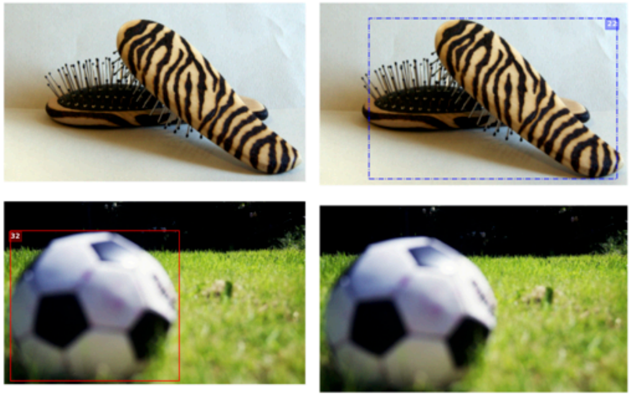

Figure 4. Images from COCO-full whose label quality is better assessed via ObjectLab than mAP score. We show both the original given label (left column in red) and prediction from Detectron2-X101 model (right column in blue). **In top row**: model incorrectly predicts hairbrush is a zebra (class #22) with moderate confidence. The label quality score for this image via mAP = 0.0, via ObjectLab = 1.0. **In bottom row**: model does not recognize sports ball (class #32) but it is correctly marked in the annotation. The label quality score for this image via mAP = 0.0, via ObjectLab = 1.0.

The images with the lowest ObjectLab quality scores in the COCO-full dataset reveal many interesting findings. Figure 1 illustrates different types of label errors present in the dataset, and Figure 6 shows fundamental inconsistencies between annotations. Through visual examination of many top/mid/bottom-ranking ObjectLab results, we estimate that in COCO 2017 around: 5% of images have an *Overlooked* error, 3% have a *Badly Located* error, and 0.7% have a *Swapped* error. A full table of ObjectLab results for the entire COCO-full dataset is provided in the previously linked benchmarks GitHub repository.

6. Discussion

The ObjectLab score introduced in this paper is straightforward to integrate into any existing object detection pipeline. Using predictions from the trained model, ObjectLab is able to detect diverse types of errors and automatically prioritizes mislabeled images in the data for review. After their labels are fixed, the same object detection model can be easily retrained on the corrected dataset. Because ObjectLab depends on model predictions, its label error detection accuracy increases with a better model. Thus a better model improves ObjectLab results, which in turn can be used to better correct the data, allowing an even better version of the model to be trained. Most object detection models and datasets should be amenable to this virtuous cycle. In practice, labeling issues should be considered in both training and evaluation datasets (Northcutt et al., 2021a), to not only maximize reliability of a learned object detector but also ensure decisions like architecture selection and whether to deploy or not are based on the correct information.

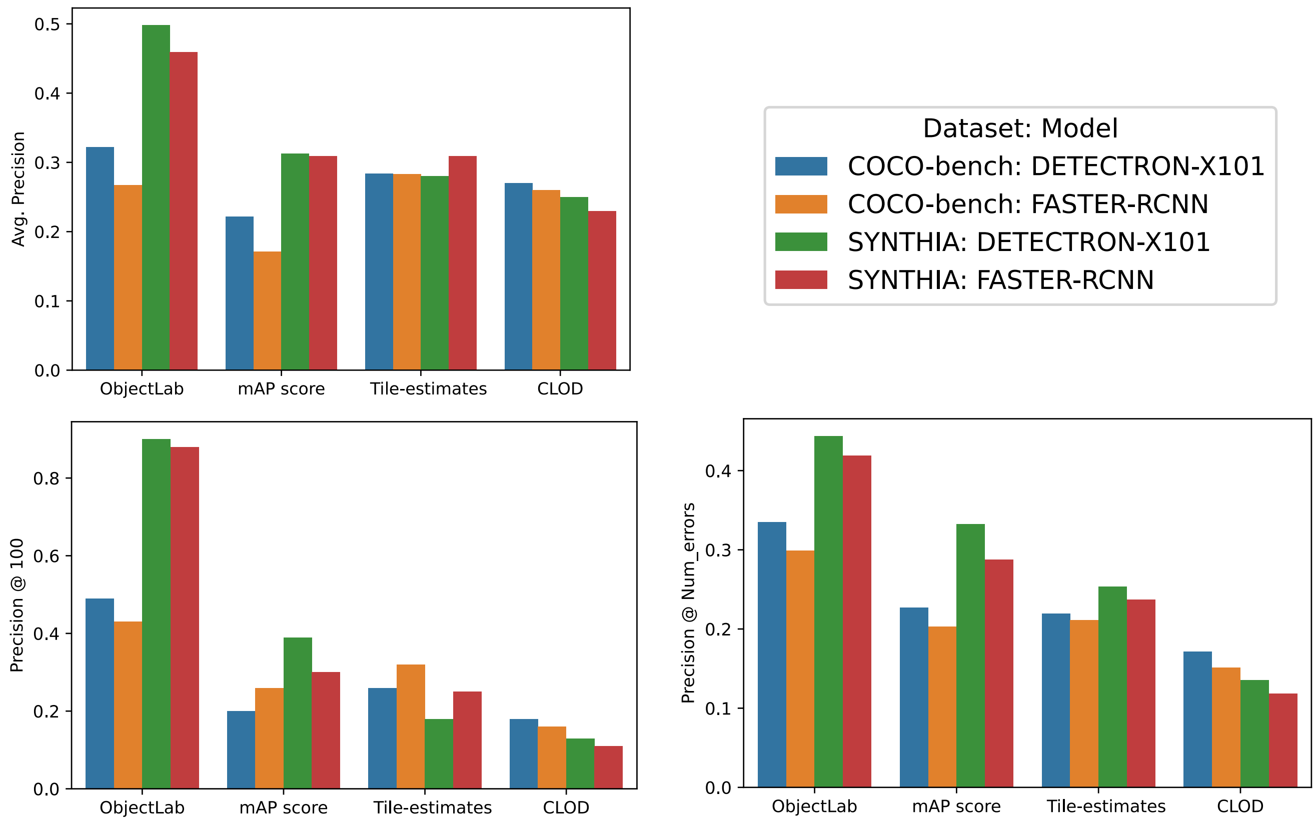

Figure 5. Evaluating various label quality scoring methods across two models and two datasets where ground truth label errors are known.

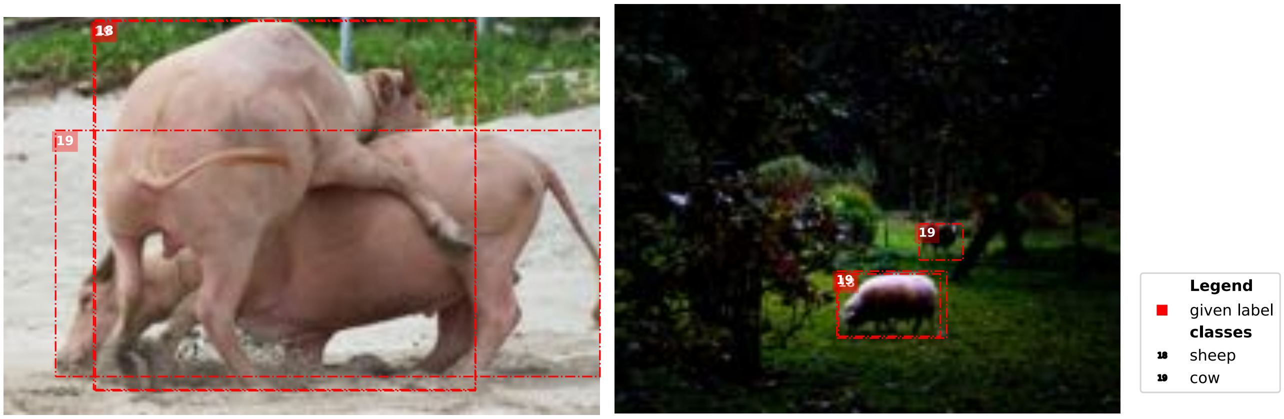

Figure 6. Examples of inconsistent annotations in COCO-full. In these images, which received low ObjectLab scores, some depicted animals are annotated as both **cow** and **sheep**.

References

Bengar, J. Z., Gonzalez-Garcia, A., Villalonga, G., Raducanu, B., Aghdam, H. H., Mozerov, M., Lopez, A. M., and Van de Weijer, J. Temporal coherence for active learning in videos. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 914–923. IEEE, 2019.

Bolya, D., Foley, S., Hays, J., and Hoffman, J. Tide: A general toolbox for identifying object detection errors. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pp. 558–573. Springer, 2020.

Brodley, C. E. and Friedl, M. A. Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167, 1999.

Chachuła, K., Popowicz, A., Łyskawa, J., Olber, B., Fratczak, P., and Radlak, K. Combating noisy labels in object detection datasets. *arXiv preprint arXiv:2211.13993*, 2022.

Chan, R., Lis, K., Uhlemeyer, S., Blum, H., Honari, S., Siegwart, R., Fua, P., Salzmann, M., and Rottmann, M. Segmentmeifyoucan: A benchmark for anomaly segmentation, 2021.

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., and Lin, D. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

Hasty.ai. How we cleaned up PASCAL and improved mAP by 13%. https://www.edge-ai-vision.com/2022/08/how-we-cleaned-up-pascal-and-improved-map-by-13/, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, 2018.

Klie, J.-C., Webber, B., and Gurevych, I. Annotation error detection: Analyzing the past and present for a more coherent future. *arXiv preprint arXiv:2206.02280*, 2022.

Kuan, J. and Mueller, J. Model-agnostic label quality scoring to detect real-world label errors. In *ICML DataPerf Workshop*, 2022.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection, 2017.

Ma, J., Ushiku, Y., and Sagara, M. The effect of improving annotation quality on object detection datasets: A preliminary study. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Muller, N. M. and Markert, K. Identifying mislabeled instances in classification datasets. In *International Joint Conference on Neural Networks*. IEEE, jul 2019.

Murrugarra-Llerena, J., Kirsten, L. N., and Jung, C. R. Can we trust bounding box annotations for object detection? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4813–4822, 2022.

Nishi, K., Ding, Y., Rich, A., and Hollerer, T. Augmentation strategies for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

Northcutt, C. G., Athalye, A., and Mueller, J. Pervasive label errors in test sets destabilize machine learning benchmarks. In *Proceedings of the 35th Conference on Neural Information Processing Systems Track on Datasets and Benchmarks*, December 2021a.

Northcutt, C. G., Jiang, L., and Chuang, I. L. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411, 2021b.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

Ros, G., Sellart, L., Materzynska, J., Vazquez, D., and Lopez, A. M. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3234–3243, 2016. doi: 10.1109/CVPR.2016.352.

Rottmann, M. and Reese, M. Automated detection of label errors in semantic segmentation datasets via deep learning and uncertainty quantification, 2022.

sama. The sama-coco dataset. https://www.sama.com/sama-coco-dataset/, 2022.

Sukhbaatar, S. and Fergus, R. Learning from noisy labels with deep neural networks. *arXiv preprint arXiv:1406.2080*, 2014.

Thyagarajan, A., Snorrason, E., Northcutt, C., and Mueller, J. Identifying incorrect annotations in multi-label classification data. *ICLR Workshop on Trustworthy ML*, 2023.

Voxel51. Finding detection mistakes with fiftyone. https://docs.voxel51.com/tutorials/detection_mistakes.html, 2023.

Wang, W.-C. and Mueller, J. Detecting label errors in token classification data. In *NeurIPS Workshop on Interactive Learning for Natural Language Processing*, 2022.

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks, 2017.

Xu, M., Bai, Y., Ghanem, B., Liu, B., Gao, Y., Guo, N., Ye, X., Wan, F., You, H., Fan, D., et al. Missing labels in object detection. In *CVPR workshops*, volume 3, pp. 5, 2019.

Zaidi, S. S. A., Ansari, M. S., Aslam, A., Kanwal, N., Asghar, M., and Lee, B. A survey of modern deep learning based object detection models. *Digital Signal Processing*, pp. 103514, 2022.

Zhang, Z. and Sabuncu, M. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems*, 2018.

Supplementary Results

Dataset	Model	mAP
COCO-bench	X-101	0.67
COCO-bench	FasterRCNN	0.62
SYNTHIA	Detectron:X-101	0.58
SYNTHIA	FasterRCNN	0.53
COCO-full	X-101	0.53
COCO-full	FasterRCNN	0.49

Table 2. Standard mAP evaluation to measure overall accuracy of the (out-of-sample) predictions from each model on each dataset.

Dataset : Model	Quality Score	Avg. Precision	Precision@100	Precision@Num_errors
COCO-bench: X-101	ObjectLab	0.365	0.49	0.34
	mAP	0.222	0.20	0.23
	Tile-estimates	0.284	0.26	0.22
	CLOD	0.27	0.18	0.17
COCO-bench: FRCNN	ObjectLab	0.273	0.43	0.30
	mAP	0.171	0.26	0.20
	Tile-estimates	0.283	0.32	0.21
	CLOD	0.26	0.16	0.15
SYNTHIA: X-101	ObjectLab	0.502	0.89	0.44
	mAP	0.313	0.39	0.33
	Tile-estimates	0.280	0.18	0.25
	CLOD	0.25	0.13	0.14
SYNTHIA: FRCNN	ObjectLab	0.46	0.88	0.42
	mAP	0.309	0.30	0.29
	Tile-estimates	0.309	0.25	0.24
	CLOD	0.23	0.11	0.12

Table 3. Metrics of various label quality scoring methods across two models and two datasets where ground truth label errors are known. This table is simply an alternate representation of the benchmark results plotted in Figure 5.

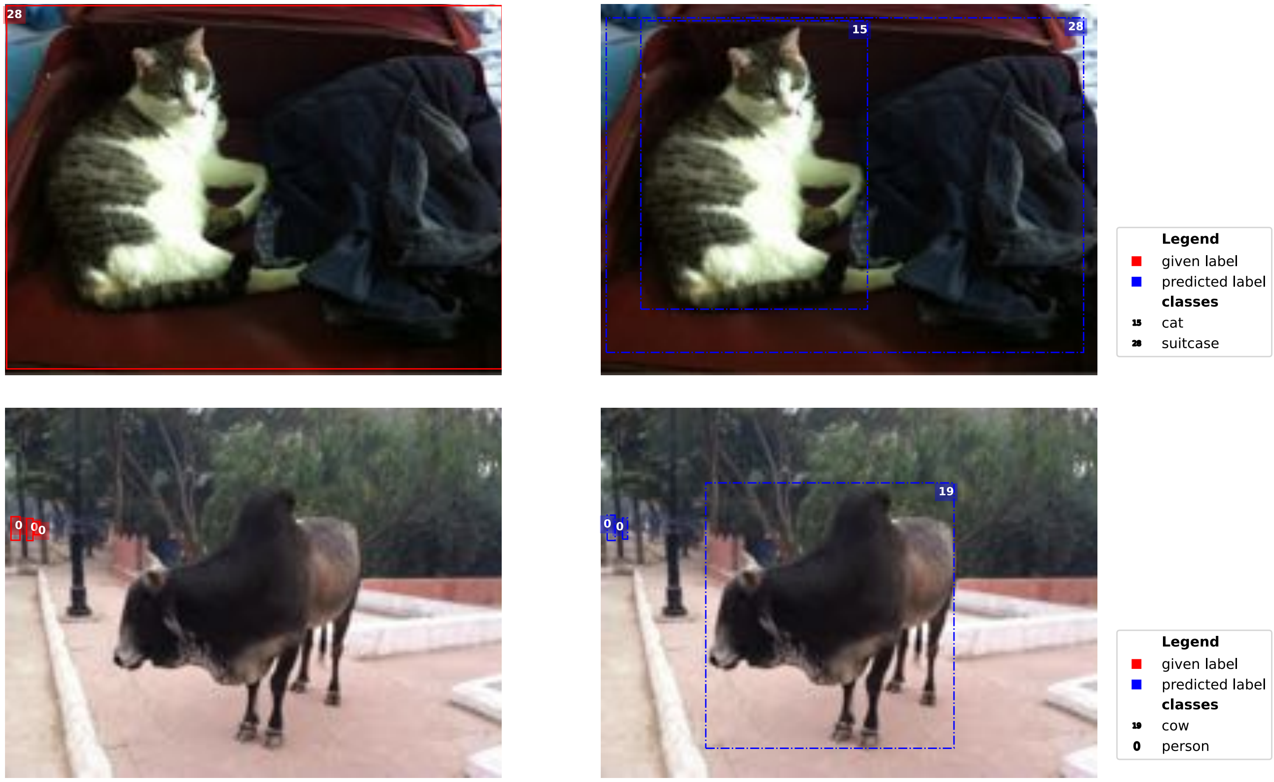

Figure 7. Additional examples of *Overlooked* errors amongst the images with lowest ObjectLab scores in the COCO-full dataset. In top row: the given label (on left) is missing the **cat** (class #28), detected by Detectron2-X101 model with confidence $= 0.99$ (prediction on right). In bottom row: an atypical-looking **cow** (class #19) was missed by annotators, but predicted by Detectron2-X101 model with confidence $= 0.99$.

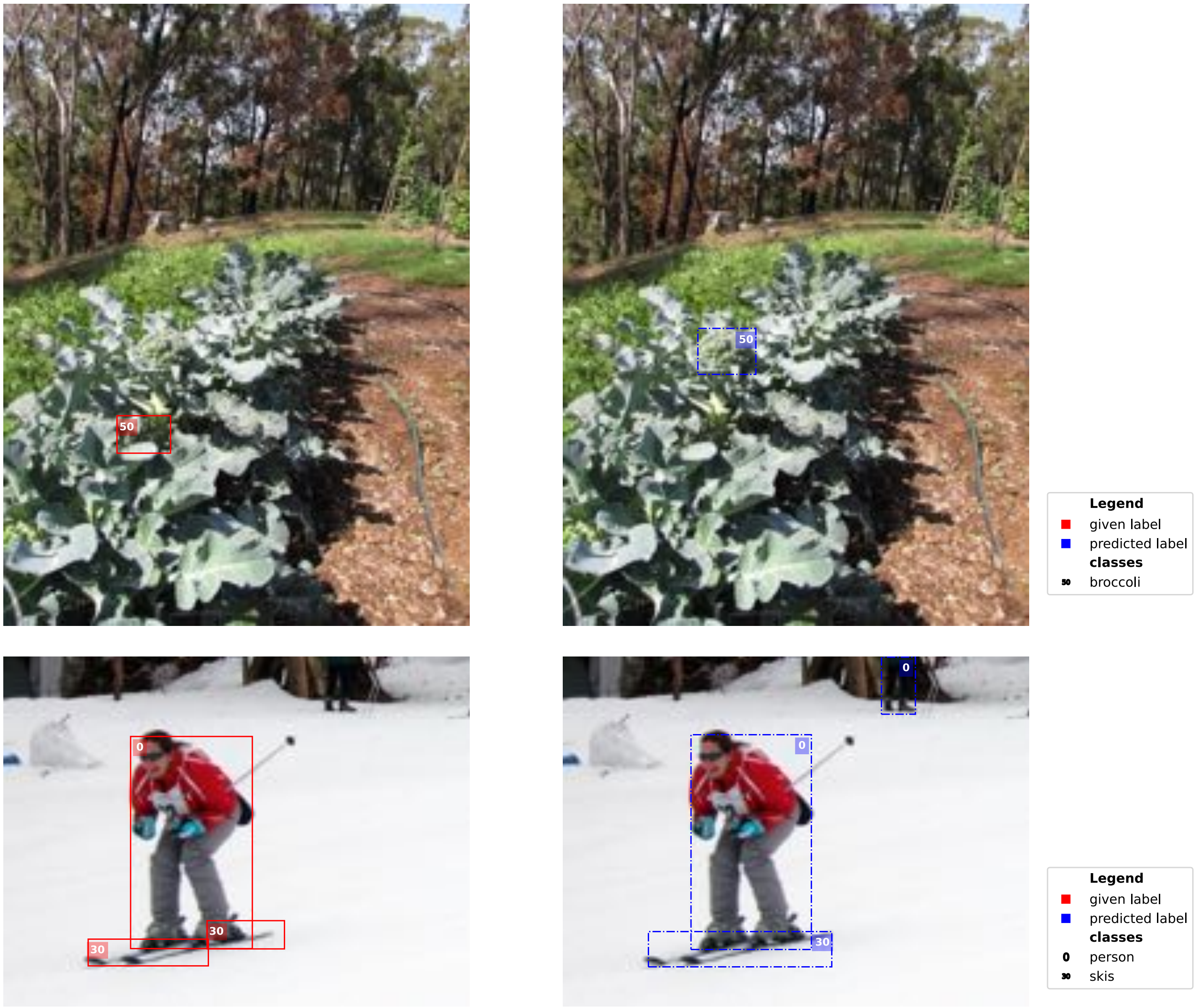

Figure 8. Additional examples of *Badly Located* errors amongst the images with lowest ObjectLab scores in the COCO-full dataset. In top row: the given label (on left) incorrectly localizes the flower head of the **broccoli**, whereas Detectron2-X101 model correctly predicts its location with moderate confidence $= 0.88$ (on right). In bottom row: the annotated bounding boxes were poorly drawn around the **skis**.

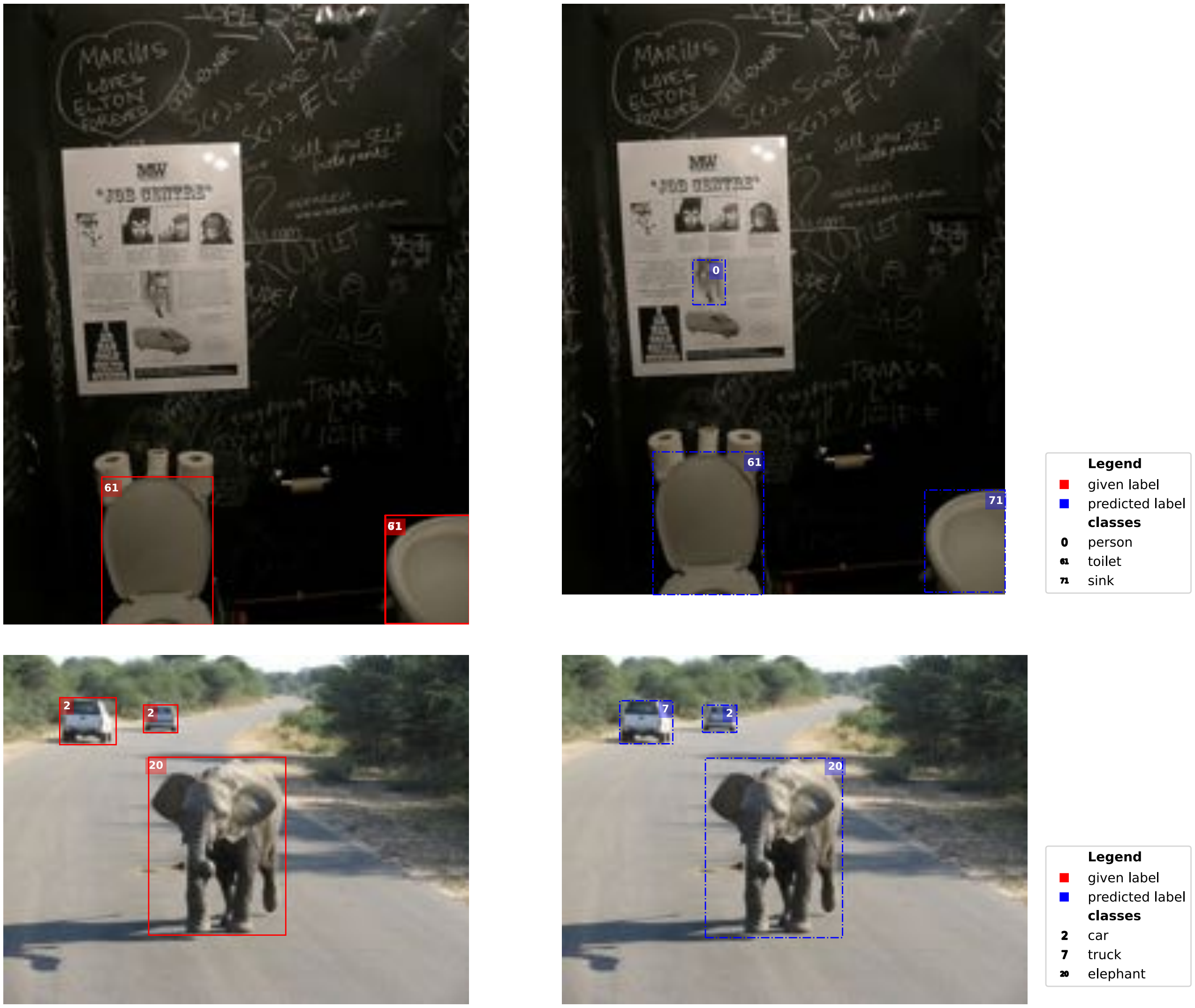

Figure 9. Additional examples of *Swapped* errors amongst the images with lowest ObjectLab scores in the COCO-full dataset. In top row: the given label (on left) mistakenly says the **sink** to the right of the toilet is another **toilet**, whereas Detectron2-X101 model confidently predicts it is a **sink** (confidence = 0.95). In bottom row: the depicted **truck** is incorrectly annotated as a regular **car** in the given label, whereas these are separate classes in COCO-full.

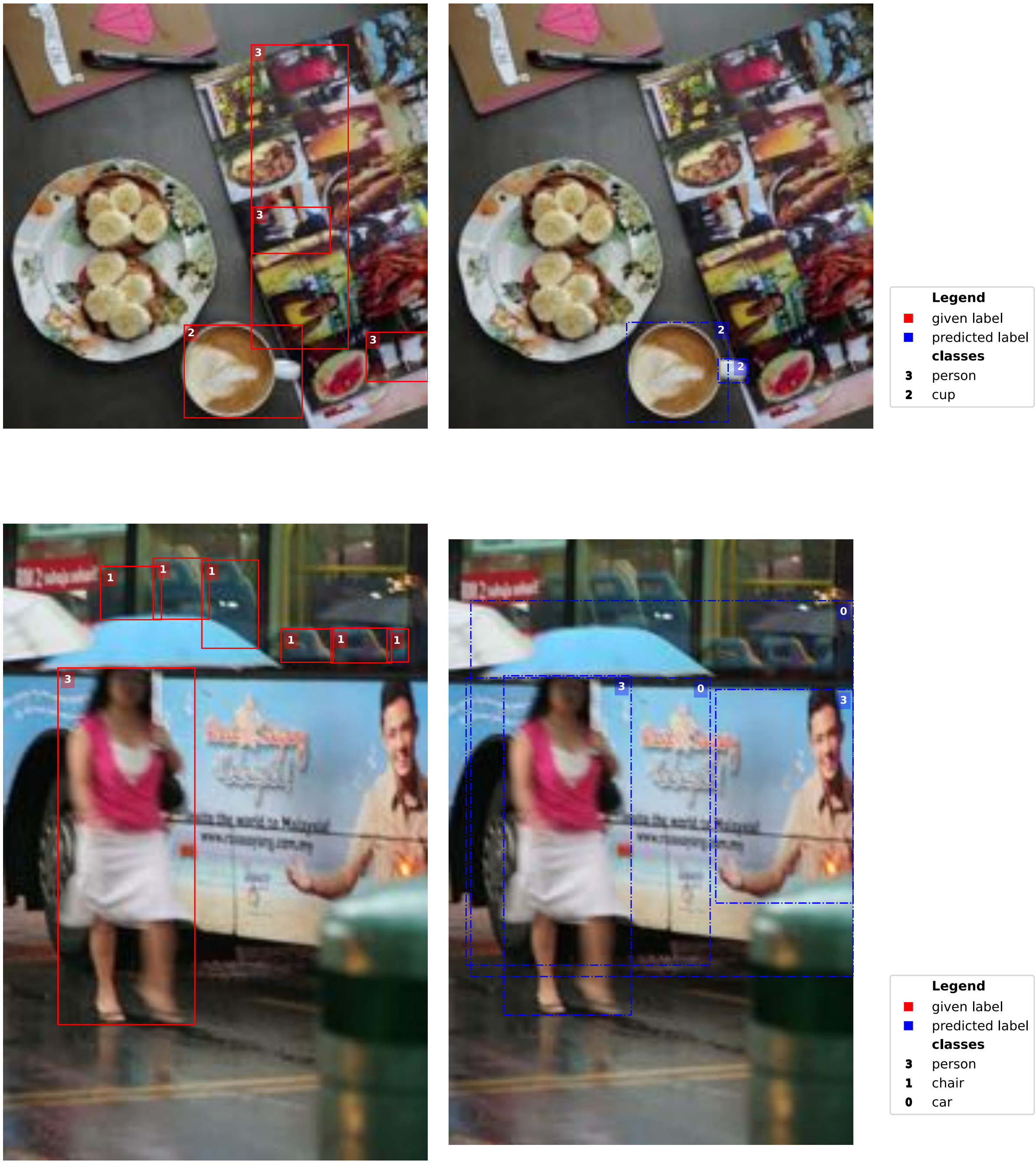

Figure 10. Examples of inconsistent annotations in COCO-full detected via low ObjectLab score (with predictions from Detectron-X101 model). Here we see selective annotation of *photos* of people as **person** objects in some images but not others.

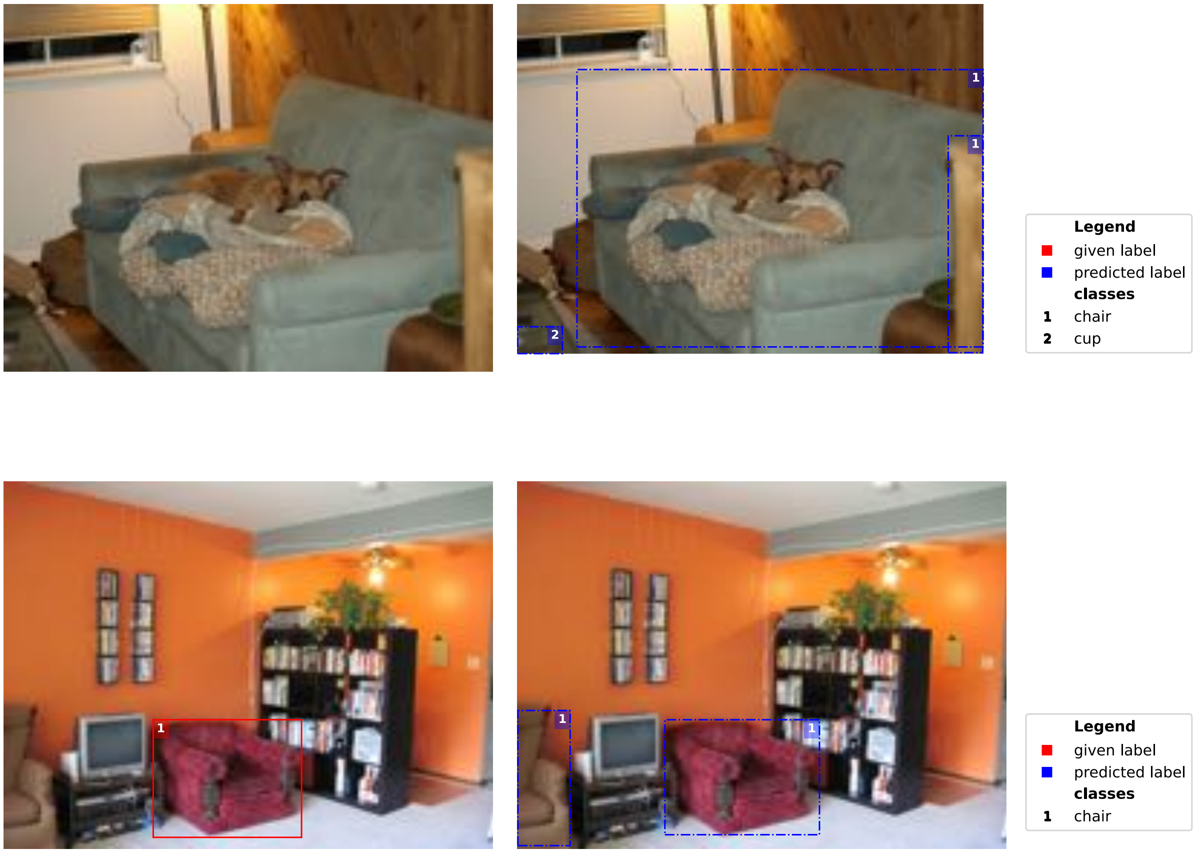

Figure 11. Examples of inconsistent annotations in COCO-full detected via low ObjectLab score (with predictions from Detectron-X101 model). Here we see selective annotation of sofas as **chair** objects in some images but not others.

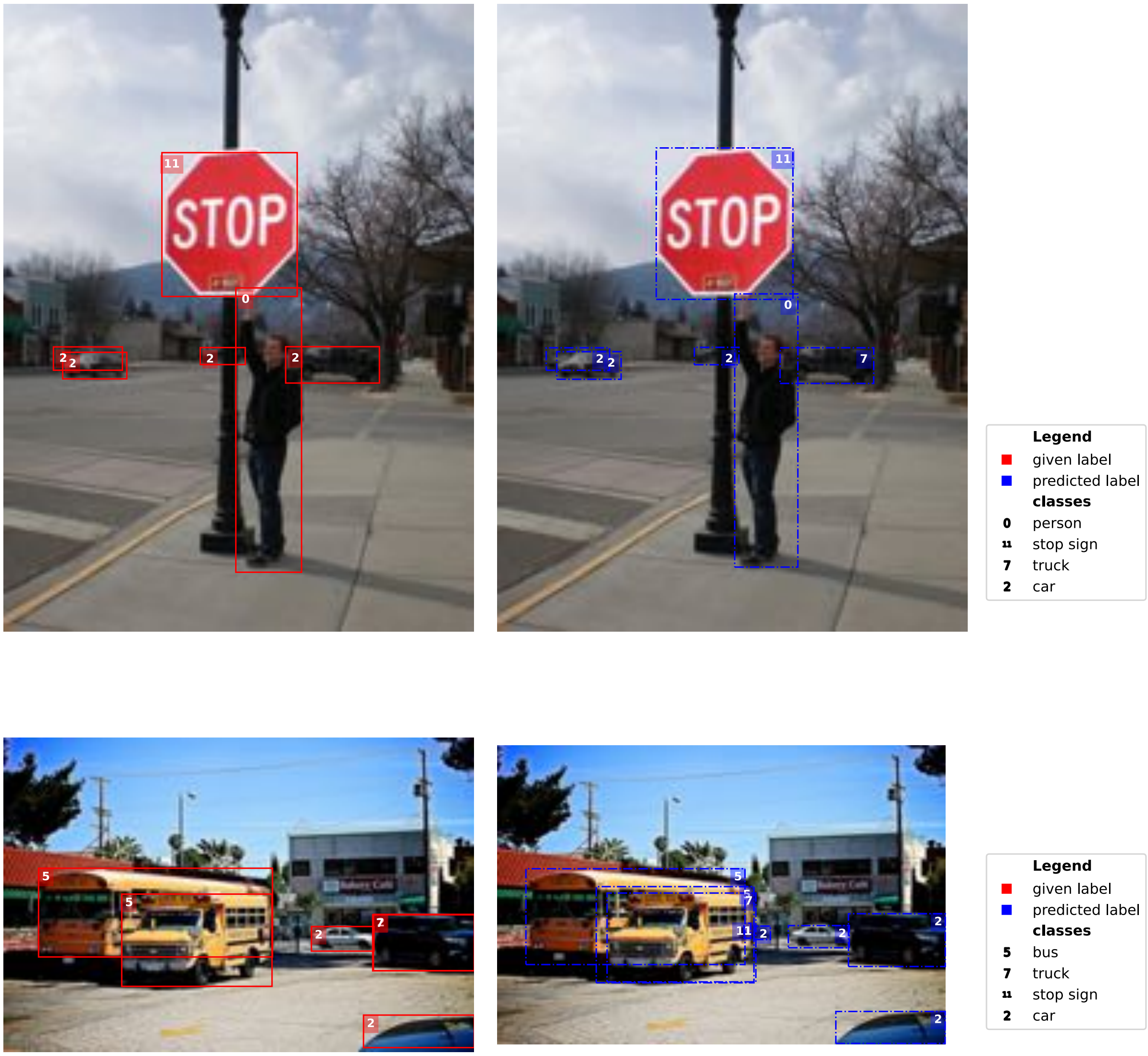

Figure 12. Examples of inconsistent annotations in COCO-full detected via low ObjectLab score (with predictions from Detectron-X101 model). Here we see selective annotation of SUVs as **car** objects in some images but as **truck** objects in others.