Evaluating the Effect of Common Annotation Faults on Object Detection Techniques

Abraham Chan, Arpan Gujarati, Karthik Pattabiraman, Sathish Gopalakrishnan
The University of British Columbia (UBC), Vancouver, BC, Canada
Email: abrahamc@ece.ubc.ca, arpanbg@cs.ubc.ca, karthikp@ece.ubc.ca, sathish@ece.ubc.ca

***Abstract*—Machine learning (ML) is applied in many safety-critical domains such as autonomous driving and medical diagnosis. Many ML applications in such domains require object detection, which includes both classification and localization, to provide additional context. To ensure high accuracy, state-of-the-art object detection (OD) systems require large quantities of correctly annotated images for training. However, creating such datasets is non-trivial, may involve significant human effort, and is hence inevitably prone to annotation faults. We evaluate the effect of such faults on OD applications. We present ODFI, which can inject five different types of common annotation faults into any COCO-formatted dataset. We then use ODFI to inject these faults into two road traffic and one medical X-ray imaging datasets. Finally, using these faulty datasets, we systematically evaluate and compare the efficacy of existing OD techniques that are designed to be robust against such faults. To do so, we introduce a new metric that evaluates the *robustness* of OD models in the presence of faults. We find that (1) single-stage detectors trained with faulty annotations perform better in scenes with more objects, (2) redundant bounding boxes have the least impact on robustness, and (3) ensembles have the highest overall robustness among the robust OD techniques considered.**

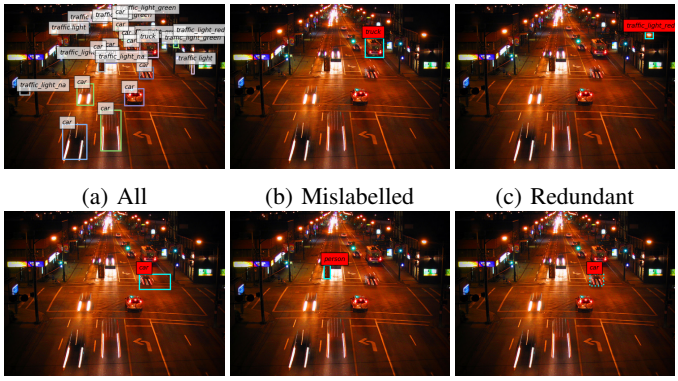
***Index Terms*—Error resilience, Machine learning, Training**

I. Introduction

Machine learning (ML) is used in many safety-critical domains such as autonomous vehicles (AVs) [1] and medical diagnosis [2]. AVs rely on ML to make sound driving decisions [3]. Medical practitioners may use ML to guide diagnosis in X-ray images [4]. Incorrect predictions could cause an AV to crash (endangering its occupants) or could lead to a misdiagnosis (placing the patient's health at risk).

Object Detection (OD), or image classification and localization of multiple objects within a single image frame, is an important ML application [5]. ML models in AVs, for instance, need to identify all pedestrians and their positions in the frame. An AV should brake if a pedestrian appears directly on its oncoming path. However, if the same pedestrian was positioned to the side of the road, the AV need not brake. Similarly, in medical diagnosis, ML models need to identify the disease type and locate the position of abnormal tissues in an X-Ray image.

OD is based largely on supervised ML [5] and requires a large collection of annotated images for training, e.g., the widely used COCO dataset [6]. To ensure high accuracy, a large fraction of annotations in the training dataset must be *correct*, *i.e.*, they must (1) classify the object classes correctly, (2) define the bounding box sizes and positions correctly, and (3) not miss any or have redundant bounding boxes.



(a) All (b) Mislabelled (c) Redundant

(d) Incorrect Box (e) Mislabelled SC (f) Missing

Fig. 1: **(a)** Image #123289 in COCO [6] with all annotations. **(b, c)** Annotation faults found in the image. **(d-f)** Annotation faults interpolated in the image (SC denotes superclass).

Popular training datasets for object detection such as COCO [6], *unfortunately*, have been found to contain numerous annotation faults. For example, Fig. 1a shows an original training image from COCO with all the annotations; Fig. 1b identifies an annotation with a *mislabelled class* in this image (a bus is incorrectly classified as a truck), whereas Fig. 1c identifies a *redundant annotation* (a street light is unnecessarily annotated as a traffic light). As per *understand.ai*, an industry expert in data annotation, *annotations with incorrect bounding boxes*, *mislabelled superclasses*, and *missing annotations* are also quite common [7]. While the original image from COCO shown in Fig. 1a does not contain these three faults, we interpolate these onto the image in Figs. 1d to 1f. The bounding box in Fig. 1d has incorrect dimensions, the traffic light in Fig. 1e is incorrectly annotated as a person (a different superclass), and the bounding box is missing in Fig. 1f.

All five fault types (Fig. 1) can lead to unsafe and potentially catastrophic scenarios (see Table I). However, object detection models are trained over thousands of annotated images, so predictions at runtime may actually not result in similar faults. The occasional fault in a dataset may not affect the overall accuracy but we sometimes do not know the extent to which training data is faulty. We therefore ask the question: *When do faulty annotations in the training dataset start affecting the*

prediction quality of an object detection model? This specific question breaks down into three related questions:

① What is the effect of faults in training data on object detection? ② Are specific types of faults more impactful than others? ③ What are effective techniques for improving the robustness of object detection?

We systematically study the effect of annotation faults on OD techniques via fault injection. We focus on the five types of annotation faults identified above. We further study the efficacy of *robust OD techniques*, which are designed to produce models that maintain high classification and localization accuracy despite the presence of different types and amounts of annotation faults in training datasets. An example robust OD technique uses a secondary ML network to distill clean training annotations from noisy ones and learns from them [8].

In our prior work, we studied the robustness of image classification models in the presence of training dataset faults [9], but not for object annotation faults on OD techniques. Unlike image classification, object detection also entails the prediction of correctly sized bounding boxes, known as localization. The process of localizing bounding box coordinates and sizes makes it a regression problem, where the output space is much larger than classification. Other studies on robust object detection have focused mostly on evaluating and tolerating the effects of salt and pepper noise, Gaussian noise, or adverse weather events on object detection training images [10–14]. Comparatively, fewer studies have focused on evaluating or tolerating the common annotation faults we described. Existing studies [8, 15, 16] focus specifically on either mislabelled class or incorrect bounding box errors, while we consider multiple fault types. *To the best of our knowledge, we are the first to systematically study the robustness of objection detection models, and the effectiveness of robust object detection techniques in the presence of commonly observed annotation faults.*

We take the perspective of a practitioner rather than a ML researcher in this work. Practitioners desire out-of-the-box (*i.e.* general) approaches in ML, with minimal tuning, such that their own datasets and models can be easily evaluated with interpretable metrics [17]. Therefore, we systematically identify representative robust OD techniques from the literature that can be easily applied with minimal effort across datasets. We also propose a new metric that is easily interpretable in the context of different safety-critical applications, and allows consistent comparison among the different OD techniques.

We make the following contributions:

- We propose a fault injection tool called Object Detection Fault Injector (ODFI[1]), which can systematically inject five different types of common annotation faults into any COCO-formatted [18] dataset, a popular standard for object detection datasets.
- We introduce a new metric, Object Precision Delta (OPD), that evaluates the robustness of ML models in safety-critical object detection by placing higher significance on mispredicted annotations belonging to incorrect superclasses. OPD provides a consistent metric across different safety-critical datasets, which have vastly different ways to measure prediction capability.
- We systematically shortlist papers on robust OD techniques, by carrying out an extensive related work survey, categorizing the techniques, and choosing representative techniques in each category for our evaluation.
- We use ODFI and OPD to evaluate the effect of annotation faults on OD techniques, as well as the effectiveness of select robust OD techniques, against different OD models, across three OD datasets covering real-world road traffic, simulated road traffic, and medical X-Rays.

Our experimental results show that (1) two-stage OD models are less robust than single-stage OD models in images with more objects, (2) redundant annotation faults have the least impact on OD models, and (3) ensembles comprised of both single-stage and two-stage detectors provide high robustness across fault types and outperform other techniques, averaging a 34% improvement in OPD over the individual models.

[1]ODFI is available at https://github.com/DependableSystemsLab/ODFI

II. Background and Motivation

To understand the effectiveness of object detection, we need to define a suitable metric that measures the quality of inference. We start by reviewing *Mean Average Precision (mAP)*, a metric that is widely used to measure the prediction accuracy of an object detection model (Section II-A). We then discuss the limitations mAP for evaluating the *robustness* of OD models. Finally, we present an example to motivate the need for looking beyond metrics like mAP (Section II-C).

A. Background: Mean Average Precision (mAP)

There are several ways to compute mAP and many online tutorials explain these steps in detail [19–21]. Our approach is based on the mAP definition proposed in the PASCAL Visual Object Classes (VOC) challenge [19, 22].

In object detection, each prediction (each object detected in an image) consists of a bounding box, a class label, and a confidence level. The first step is to determine if a prediction is a *true positive (TP)* or a *false positive (FP)*. A prediction is a TP if its class label matches the ground truth label and its bounding box overlaps reasonably well with the ground truth bounding box. The latter condition is evaluated by checking that the *Intersection-of-Union (IoU)* between the two bounding boxes, *i.e.*, the ratio of their overlapping region and their union, exceeds a minimum threshold, which is typically 0.5 [23]. Any prediction that does not satisfy the TP criteria is a FP. In addition, if a labeled object in the ground truth data is not predicted at all, we count it as a *false negative (FN)*.

The second step is to determine the *precision* and *recall* values for each class. Each prediction is associated with a confidence score. Given a class, all predictions identifying objects belonging to this class are first ranked in decreasing order of their confidence scores. The precision and recall values are then computed at each rank, starting from the highest rank. Precision is the fraction of correct predictions

TABLE I: List of annotation faults explored in this paper and their potential implications, based on examples in Fig. 1

Fault Type	Unsafe or Potentially Catastrophic Scenarios
Mislabelled Class	Fig. 1b: Misclassifying a bus as a truck may cause an AV to ignore bus priority lane merging rules
Redundant Annotation	Fig. 1c: The redundantly annotated red traffic light may mislead an AV to brake, overriding the (real) green light
Incorrect Bounding Box	Fig. 1d: Due to the very wide bounding box, an ML model may erroneously learn to avoid passing cars even when safe
Mislabelled Superclass (SC)	Fig. 1e: Misclassifying a random small object as a person may cause an AV to dangerously pause in an intersection
Missing Annotation	Fig. 1f: The AV may entirely disregard the car on the road (which was not annotated) causing a head-on collision

among all predictions evaluated, and recall is the fraction of ground truth predictions that have been correctly predicted.

Formally, let $OD_c = \{od_{c,1}, od_{c,2}, \ldots\}$ denote the ordered set of predictions for class c. Let $TP_?(od_{c,i}) = 1$ (or 0) if the i^{th} prediction is a TP (or a FP, respectively). Let N_c denote the total number of ground truth predictions. The precision and recall at rank r are defined as follows:

$$P_{c,r} = \frac{\sum_{i=1}^{r} TP_?(od_{c,i})}{r} \text{ and } R_{c,r} = \frac{\sum_{i=1}^{r} TP_?(od_{c,i})}{N_c}.$$

In the final step, first the *Average Precision (AP)* for each class is derived by computing the area under the *precision-recall (P-R) curve*, *e.g.*, the $P_{c,r}$ versus $R_{c,r}$ curve for class c. A step function upper-bounding the P-R curve may be used, so that AP can be easily computed as the sum of all rectangular blocks under the curve. The mAP is then computed by averaging the AP across all classes in the dataset. It ranges from 0 to 1, where higher values imply better accuracy.

B. Limitations of mAP

We identify three main issues with mAP.

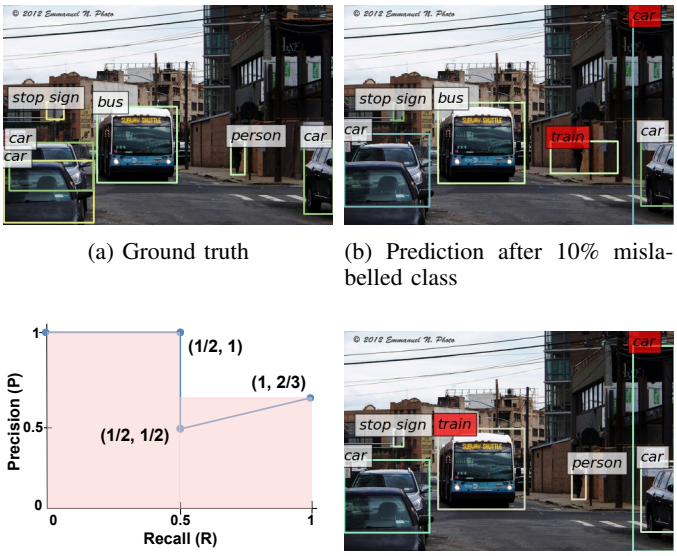
First, each incorrect prediction (*i.e.*, false positive) in an image is given the same weight regardless of its class, which is not suitable for safety-critical applications. For example, misclassifying a truck as a bus has negative safety consequences for AVs (since buses have more frequent stops than trucks); however, misclassifying a truck as a green traffic light is much worse (e.g., the AV may mistakenly drive into the truck!). These differences are not captured by the mAP metric.

Second, there are multiple definitions of mAP being used that are inconsistent, which makes comparisons across datasets difficult [24]. For example, unlike the PASCAL VOC definition that we discussed earlier, the mAP for COCO [6] is averaged across both label classes and different IoU thresholds.

Third, in the context of fault injection, mAP is oblivious of the difference between a model that is trained on data injected with faults (a *faulty* model) and a model that is trained on pristine data (*golden* model). Hence, when using mAP to evaluate a model's performance with and without fault injection, the cases where both the golden model and the faulty model incorrectly predict objects are unnecessarily included – a similar problem also arises in the accuracy metric used for classification as our prior work found [25]. Scenarios that are incorrectly predicted by the golden model should instead be filtered out and should not be used when evaluating robustness.

C. Motivating Example

We demonstrate the limitations of mAP when trying to understand the effect of annotation faults against OD models.



(a) Ground truth

(b) Prediction after 10% mislabelled class

(c) P-R Curve for "Car"

(d) Hypothetical Prediction

Fig. 2: Prediction on a test image from COCO-Traffic.

We use the COCO-Traffic [26] dataset, a subset of MS-COCO [6] containing traffic-related images.

First, we train YOLOv3 with a DarkNet-53 backbone for object detection using a pristine version of the COCO-Traffic dataset (*i.e.*, without introducing any annotation faults). The resulting model is called the *golden model*. The mAP of the golden model is 0.43, which is similar to the state-of-the-art mAP observed for YOLOv3 on MS-COCO [27]. Next, we train the same network on a different version of COCO-Traffic, this time after injecting annotation faults with mislabelled classes (similar to the real-world examples shown in Fig. 1b). Specifically, we randomly select 10% of object annotations across the entire training dataset and perturb their class categories. As a result, the mAP of the *faulty* model falls slightly from 0.43 to 0.39.

In Fig. 2a, we show a test image from COCO-Traffic overlaid with the ground truth bounding boxes and their class labels (without fault injection). In Fig. 2b, we show the same image, but overlaid with the predictions made by the faulty model above (*i.e.*, which was trained on COCO-Traffic injected with 10% mislabelled class). As a result, some predictions in Fig. 2b (marked in red) are incorrect and do not match the ground truth. Despite the faulty model making many mispredictions throughout the test dataset (*e.g.* a person is misclassified as a train!), mAP only reports a small

TABLE II: AP statistics for Fig. 2b

Class	Predicted	IoU	Conf	Result	P	R	AP
Bus	Bus	0.87	0.83	TP	1/1	1/1	1
	Car	0.73	0.7	TP	1/1	1/2	
Car	Car	0	0.63	FP	1/2	1/2	0.83
	Car	0.6	0.47	TP	2/3	2/2	
Stop sign	Stop sign	0.5	0.36	TP	1/1	1/1	1
Person	-	-	-	FN	0/0	0/1	0
Train	Train	0.21	0.51	FP	0/1	0/0	0

TABLE III: AP statistics for Fig. 2d

Class	Predicted	IoU	Conf	Result	P	R	AP
Bus	-	-	-	FN	0/1	0/0	0
Car	Car			*no changes*			0.83
Stop sign	Stop sign			*no changes*			1
Person	Person	0.9	0.51	TP	1/1	1/1	1
Train	Train	0.87	0.83	FP	0/1	0/0	0

decrease from the golden model, showing its unsuitability in comparing robustness. To understand the shortcomings of mAP, we explain how mAP is computed on a single image, and how it changes due to mispredictions by a faulty model.

We list all the predictions in Fig. 2b in Table II. The predictions are grouped by class, and in the case of multiple predictions per class, ordered by their confidence values. We mark each prediction as a true positive (TP) or a false positive (FP) depending on their IoU values and class labels. If an object in the ground truth is not detected (*e.g.*, class "Person"), we mark it as a false negative (FN).

To compute class-wise APs, we follow the steps in Section II-A. We compute the precision (P) and recall (R) of each prediction in order of their rank, and then plot the P-R curve (see Fig. 2c). Using rectangular interpolation [19, 22] shown as shaded, we compute the area under the P-R curve to yield the AP. The mAP, averaged over the five AP values, one for each class, is 0.57. While mAP is flexible over imperfectly predicted bounding boxes (reducing FNs), mAP unfortunately treats all mispredictions equally - we show why this is a problem below.

Suppose the person in Fig. 2a was predicted correctly, while the bus was instead misclassified as a train, as shown in Fig. 2d. In Table III, we show how mAP would be recalculated. mAP would be the same (0.57), as in Fig. 2b. However, because busses and trains are both large vehicles, we expect that the relative impact in Fig. 2d would be less than that in the case where a person is mispredicted as a train (Fig. 2b).

III. Methodology for Assessing Object Detection

We define a new metric for robustness, which we refer to as *Object Precision Delta (OPD)* (in Section III-A). We survey and present three representative robust object detection techniques for evaluation (Sections III-B and III-C). Finally, we explain our fault injection tool ODFI (Section III-D).

A. A Superclass-Aware OD Robustness Metric

Like mAP, OPD also relies on the area under the precision-recall curve that is calculated over the entire test dataset. However, it incorporates other properties, as explained below.

Recall the ground truth, golden model, and faulty model definitions from Section II-C. The ground truth is the collection of labelled bounding boxes in the test dataset, the golden model is trained on the training dataset before fault injection, and the faulty model is trained on the training dataset after fault injection. Unlike mAP, OPD has the following properties.

1) When OPD is computed on faulty models, it excludes any object that is incorrectly predicted by the golden model. This eliminates the double-counting problem with mAP identified in Section II-B.
2) OPD compares the superclass of the object class predicted by the faulty model with that predicted by the golden model. It then uses weighted FPs so that a FP that mispredicts the superclass is penalized more than a FP that mispredicts only the class. In contrast, mAP ignores the superclass and treats all FPs equally.

We now formally define OPD. Consider a labelled dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X}$ and $\mathcal{Y}$ denote an ordered set of images and their labels. Let $x_i \in \mathcal{X}$ denote the i^{th} input image and $y_i \in \mathcal{Y}$ denote the set of all annotations associated with it. Let M_g and M_f denote two trained instances of the same network architecture. M_g is the golden model and is trained on a pristine copy of the dataset, whereas M_f is the faulty model and is trained on a faulty version of the dataset. $M_g(x_i)$ and $M_f(x_i)$ denote the predictions of the two models for input image x_i. Consider another dataset $\mathcal{D}' \subseteq \mathcal{D}$ consisting of test inputs and label pairs that are correctly predicted by M_g. That is, $\mathcal{D}' = \{\mathcal{X}', \mathcal{Y}'\}$ such that $\mathcal{X}' = \{x_i \in \mathcal{X} \mid M_g(x_i) \equiv y_i\}$ and $\mathcal{Y}' = \{y_i \in \mathcal{Y} \mid x_i \in \mathcal{X}'\}$. The equivalence relation "$\equiv$" denotes that the prediction is correct, *i.e.*, the predicted class matches the ground truth and the IoU exceeds 0.5.

Let $\mathcal{C}$ denote the set of all classes and $\mathcal{C}_{super}$ denote the set of all superclasses for labels in $\mathcal{D}$. We define two functions: $\Gamma : \mathcal{Y} \mapsto \mathcal{C}$ and $\Psi : \mathcal{Y} \mapsto \mathcal{C}_{super}$. Given an annotation, Γ returns its class, while Ψ returns its superclass. OPD uses weighted FPs when computing the precision for any class. where the weights depend on whether the annotation matches the superclass. Specifically, we choose two parameters α and β to differentiate the two scenarios, and define the precision P_i for the i^{th} prediction, ordered by confidence score, as

$$\mathrm{P_i} = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + w(y_g, y_p)\mathrm{FP}_i}, \quad \text{where} \tag{1}$$

$$w(y_g, y_p) = \begin{cases} \alpha & \Gamma(y_g) \neq \Gamma(y_p) \wedge \Psi(y_g) = \Psi(y_p) \\ \beta & \Gamma(y_g) \neq \Gamma(y_p) \wedge \Psi(y_g) \neq \Psi(y_p) \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

y_g, y_p are the ground truth and predicted annotations respectively. We choose $\alpha < 1 < \beta$ so that misclassified superclasses are penalized more than misclassified classes. The weight is equal to 1 if the object class is not assigned to any superclass.

We demonstrate how we calculate OPD over a single image, using the example shown in Table II. For further simplification, we calculate OPD for the "car" class only.

The original mAP metric is unweighted, meaning $\alpha = \beta = 1$ in Eq. (2). The corresponding P-R curve is the green curve

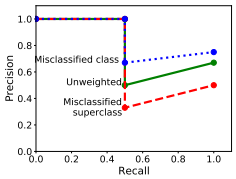


Fig. 3: P-R curves for "car". Green line is unweighted. Red dashed line shows misclassified to another superclass. Blue dotted line shows misclassified but within same superclass.

shown in Fig. 3. Before calculating the OPD, we experimented with many different combinations of α and β. If $\alpha << \beta$, OPD would only treat incorrect superclass as FPs, ignoring other cases of misprediction, while $\alpha = \beta$ would not impose any weight. We find that $\alpha = 0.5$ and $\beta = 2$ for OPD strikes this balance - hence, we use these values throughout the paper.

Suppose the second "car" annotation is misclassified as an incorrect superclass. This would result in $w(y_g, y_f) = 2$, which would lower the precision as seen by the red scatter point. We can observe that the area under the P-R curves would decrease under the red dashed curve compared to the green curve, resulting in a smaller OPD. However, if the second car annotation is mispredicted as a class belonging to the same superclass, $\alpha = 0.5$ would be applied to the FP weight, which would raise the precision (the blue dotted curve in Fig. 3).

The OPD of the golden model (YOLOv3-DarkNet) over the entire COCO-Traffic test dataset, is 0.49. After injecting 10% mislabelled class faults, the OPD drops to 0.34. This drop is greater than the drop in mAP between 0.43 to 0.39. This indicates that the YOLOv3-DarkNet model is susceptible to mispredicted superclasses, when trained with mislabelled classes - this is an observation that the mAP metric could not capture. Therefore, we define *robustness* as the difference between OPD values of the golden and faulty models.

B. Shortlisting of Robust OD techniques

To shortlist representative robust OD techniques for comparison, we followed a three step process. First, we used the following keywords to search for papers in IEEE Xplore, arXiv, and Github: "robust object detection", "noisy object detection", and "robust object detection annotation errors". This gave us 30 papers. Then, we analyzed each of the papers to check if the proposed methods are compatible with OD rather than other forms of ML (*e.g.*, image classification). Second, we categorized the papers into three main categories: ensembles, active learning, and robust loss. After categorization, we identified nine major papers (three in each category).

Finally, we chose a representative technique from each category, as one that satisfies *all the following five criteria*. (1) has available and easily modifiable code, (2) has been evaluated on more than one OD neural network architecture type, (3) has been evaluated on more than one OD dataset, (4) can read datasets in COCO Format, (5) does not significantly reduce the mAP of a fault-free model.

Table IV shows the top three papers for each category, along with how they satisfy our selection criteria. Techniques that met all selection criteria are marked using an asterisk (*); these are representative of the respective OD approach category.

C. Selected Robust OD Techniques

We explain the three techniques that are representative of each category of robust OD technique shown in Table IV.

Ensembles: Ensembles are widely used in image classification [25, 34] and also in OD [28, 30, 35]. Ensembles involve multiple models, trained independently on the same faulty training data, and voting on predictions. However, voting is not straightforward for OD since each image has multiple annotations, where each is associated with a bounding box. Because bounding box prediction is a regression problem, their exact coordinates can vary, even among golden models. Hence, a proper voting scheme must be determined for OD ensembles.

Voting schemes for ensembles [28] include: *affirmative*, *consensus*, and *unanimous*. Affirmative ensembles predict a bounding box if at least one model predicts the box. Alternatively, unanimous ensembles predict a bounding box only if every model predicts the same box. Consensus ensembles, present a middle ground, where a bounding box is predicted if it receives a majority vote across models. Consensus ensembles provide the best trade-off between false positives and false negatives [28], and hence, we implement consensus ensembles.

One advantage of ensembles over other techniques is that they can combine different OD architectures. For instance, an ensemble can combine both one-stage and two-stage detectors, and hence achieve the efficiency of single-stage detectors as well as the higher accuracy of two-stage detectors [36].

Active Learning (AL): Active learning is where the ML algorithm actively queries the training data for relevant instances, and trains on these selected instances to more effectively learn. We use the tool, OA-MIL [8], which stands for Object-Aware Multiple Instance Learning, to represent active learning. OA-MIL generates higher-quality annotations by optimizing a heuristic comparing select annotations against noisy annotations. The authors of OA-MIL evaluate it against box noise, which randomly applies incorrect bounding box transformations to every annotation in the training dataset. In contrast, we apply five different types of annotation faults, and to a random subset of the annotations in the training dataset.

Robust Loss (RL): Loss functions measure the distance between the ground truth and the predictions. In OD, there are multiple loss functions as OD consists of both classification and localization of the bounding boxes. For classification, Cross Entropy (CE) [37] is typically used as a loss function. For localization, IoU Loss [23] is commonly used.

We use Focal Loss (FL) [32] (Eq. (4)), which adds a modulating term to CE (Eq. (3)), so that learning focuses on misclassified examples. While FL is designed for class-imbalanced datasets, it may also be used for noisy annotations in OD [33]. p_c represents the prediction probability for each class, while α and γ are tunable hyperparameters. We use the default hyperparameters for FL [32] where $\alpha = 0.5$ and $\gamma = 2$.

TABLE IV: Top three techniques for each robust object detection category. Representative techniques marked with an asterisk.

OD Approach	Technique	Code?	Arch-Agnostic?	Dataset-Agnostic?	COCO-Format?	Baseline mAP?
Ensemble	Object Detection Ensemble* [28]	✓	✓	✓	✓	✓
	NoteRCNN [29]	✗	✗	✓	✓	✗
	Fused Ensemble [30]	✗	✓	✗	✗	✓
Active Learning	OA-MIL* [8]	✓	✓	✓	✓	✓
	Co-Teaching [15]	✗	✗	✓	✗	✗
	Context-aware Noisy Label Detection [31]	✗	✗	✗	✓	✓
Robust Loss	Focal Loss* [32]	✓	✓	✓	✓	✓
	Noise Resistant Focal Loss [33]	✗	✗	✓	✗	✓
	Gradient Reconcilement [16]	✓	✓	✓	✗	✓

TABLE V: Annotation fault implementation in ODFI

Fault Type	Implementation
Incorrect Box	Randomly reposition box and reduce size by 30%
Mislabelled	Replace class with another one
Mislabelled SC	Replace class with one belonging to another superclass
Missing	Remove annotation
Redundant	Duplicate annotation and position it randomly

TABLE VI: Object detection datasets used in our experiments

Name	Dataset Size		Task (# Classes)
	Training	Test	
COCO-Traffic [26]	11,544	2,887	Traffic-related images (15)
VinChest [41]	3,296	1,098	Medical Chest X-Rays (14)
CARLA [39]	1,600	264	Scenes from AV sim (10)

$$\mathrm{CE}(p_c) = -\log(p_c) \tag{3}$$

$$\mathrm{FL}(p_c) = -\alpha(1-p_c)^{\gamma}\log(p_c) \tag{4}$$

D. ODFI: Annotation Fault Injection

We explain how ODFI injects the five annotation fault types, described in Section I. ODFI has three steps. First, ODFI reads a training dataset, styled in the COCO-Format [18] into memory, using the open source COCO Dataset API [18]. Then, ODFI injects faults over a percentage of total object annotations in the training dataset. Faults of each type are injected according to their respective descriptions in Table V. For mislabelled SC, ODFI relies on superclass information provided in the original training datasets. For fault types such as *incorrect box* and *redundant*, ODFI ensures that all randomly positioned boxes are located within the image bounds. Finally, the modified dataset in memory is saved to a new COCO-Format file. While ODFI only works with datasets in COCO-Format, there are many tools [38] to convert OD datasets in other formats to the COCO-Format. The fault injection amount and fault type is easily configurable in ODFI.

IV. Evaluation

We ask the following three research questions (RQs):

- What is the robustness of object detection (OD) models without any protection technique? (**RQ1**)
- Which fault types have the greatest impact on OD robustness? (**RQ2**)
- Which OD protection techniques offer the highest robustness over the unprotected models? (**RQ3**)

A. Experimental Setup

Datasets: Table VI shows the datasets used, representing applications in AVs and medical diagnosis, which are safety-critical. We use COCO-Traffic [26] (Section II-C). The CARLA [39] dataset compiles a list of urban scenes from the CARLA AV simulator [40]. While COCO-Traffic offers real-world images of objects in traffic scenes, CARLA provides more interactive traffic scenarios between objects, which are difficult to capture in the real-world. VinChest [41] consist of Chest X-rays, which contain a collection of localized thoracic abnormalities. We rely solely on superclass information, provided by the datasets. Because VinChest does not have superclasses, we omit it from mislabelled superclass faults.

Models We use three OD models, as listed in Table VII, which cover both single-stage and two-stage detectors. Of the three OD models we used, Faster R-CNN [42] is a two-stage detector, while the YOLO-based models [27] are single-stage.

An OD model consists of a head, backbone, and sometimes a neck [43]. The backbone extracts features, while the head completes prediction by deciding which bounding boxes are relevant. The neck is a transition between the backbone and head (a Feature Pyramid Network), to increase accuracy.

We use multiple backbone models as shown in Table VII. MobileNet has about 27 convolution layers, while ResNet50 and DarkNet53 both have more than 50 convolution layers each. All backbone models are pre-trained with ImageNet weights as the standard practice in OD [44]. All unprotected OD models utilize cross entropy loss.

A two-stage detector separately conducts feature extraction, and then identifies areas of interest in two steps, while a single-stage detector combines both steps into one. For example, two-stage detectors, like Faster R-CNN, use a separate Region Proposal Network (RPN) as their heads. Two-stage detectors are typically more accurate (*i.e.* higher mAP) than single-stage detectors, but are often slower. As a result, two-stage detectors are typically preferred in medical diagnosis applications where accuracy is more important than speed, while single-stage detectors are preferred in real-time systems like AVs, where speed is more important. We choose YOLOv3 [27] with

TABLE VII: Object detection models used in our experiments

Name	Head	Neck	Backbone	# Stages
YOLOv3 MobileNet [27]	YOLOv3	-	MobileNet	Single
YOLOv3 DarkNet [27]	YOLOv3	-	DarkNet53	Single
Faster R-CNN [42]	RPN	FPN	ResNet50	Two

varying backbones to represent OD systems deployed in AV object, lane, and pothole detection systems [45, 46]. Despite the release of newer single-stage models, YOLOv3 remains an appealing choice for AVs due to its fast inference speed [47]. We apply Faster R-CNN to represent OD systems deployed for cancer and lesion detection on radiographs [48, 49].

Techniques For the robust OD techniques, we only apply Active Learning (AL) and Robust Loss (RL) to Faster R-CNN, the best performing OD model by OPD. The ensemble consists of all three individual models (Table VII) combined, under consensus voting (Section III-C). We use the default hyperparameters provided by each shortlisted technique.

Setup All the models used in our experiments are written in either TensorFlow 2.10.1 or PyTorch 1.10, trained with 100 epochs and early-stopping enabled. They are run 20 times each to obtain results within a 95% confidence interval, totalling a month of computation time. ODFI is built on the official COCO API [6], and is also written in TensorFlow. For experiments, we used a 64-bit, AMD Ryzen Threadripper 3960X 24-Core Processor and a NVIDIA RTX 3070 GPU.

B. Baseline mAP without Faults

We first measure the baseline mAP of each model and each robust OD technique without any faults injected in the training dataset. These values are shown in Table VIII, which are consistent with the state-of-the-art mAP reported for each dataset: MS-COCO [27, 42], CARLA [50], and VinChest [51].

We observe that Faster R-CNN has a higher mAP than the two YOLO-based OD models on CARLA and VinChest. However, Faster R-CNN has a lower mAP for COCO-Traffic. We find that YOLOv3-based models perform better in images containing more objects, such as COCO-Traffic. As a two-stage detector, Faster R-CNN utilizes a RPN, which yields more weight to lower confidence bounding boxes. Since YOLO is single-stage, it scans over the image only once, mitigating the impact of learning low-confidence objects.

Among the OD techniques and across datasets, we observe that ensembles have the highest overall mAP. However, they are outperformed by AL on COCO-Traffic. AL can potentially improve learning in general, even in datasets without noise, by boosting the class discriminative ability of the OD model through better learning examples [8]. We also observe that RL has the lowest overall baseline mAP across the datasets. Focal Loss, used in RL, has lower classification precision, in datasets containing less labelling noise or more balanced classes [16].

C. RQ1: Robustness without Protection

We explore the robustness of OD models without any protection techniques applied. First, we inject one fault of each fault type into 0%, 10% and 30% of the training images, and measure the mAP and OPD of each configuration. We repeat this experiment multiple times to get a 95% confidence interval. Fig. 4 shows the average mAP of YOLOv3-MobileNet for each fault type on the COCO-Traffic dataset. We observe no significant difference in mAP. We make similar observations for OPD, and on other datasets. Thus, by injecting one fault per image, the number of faulty annotations are too small to have an impact. For example in COCO-Traffic, there are 11,544 training images, containing 83,609 annotations since each image can have multiple annotations. If 10% of images are injected with one fault each, the effective faulty object rate is only $1154/83609 = 1.4\%$. Moreover, training images usually have multiple annotation faults [52, 53]. For example, an analysis of the full COCO Dataset [6] found that 37% of all object annotations were faulty [54]. *Therefore, we inject faults over the percentage of total object annotations in each dataset, rather than just a single fault per image.*

TABLE VIII: mAP of OD model accuracies trained without injected faults. The highest mAP for each dataset is bolded.

Model/Technique	Dataset		
	COCO-Traffic	CARLA	VinChest
YOLOv3 MobileNet	43.5	75.16	19.39
YOLOv3 DarkNet	43.26	64.03	19.63
Faster R-CNN	34.56	74.09	23.19
Ensemble (Ens)	42.45	**75.42**	**23.34**
Active Learning (AL)	**44.25**	74.48	21.72
Robust Loss (RL)	30.61	65.79	21.76

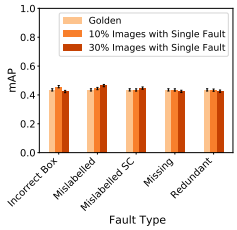
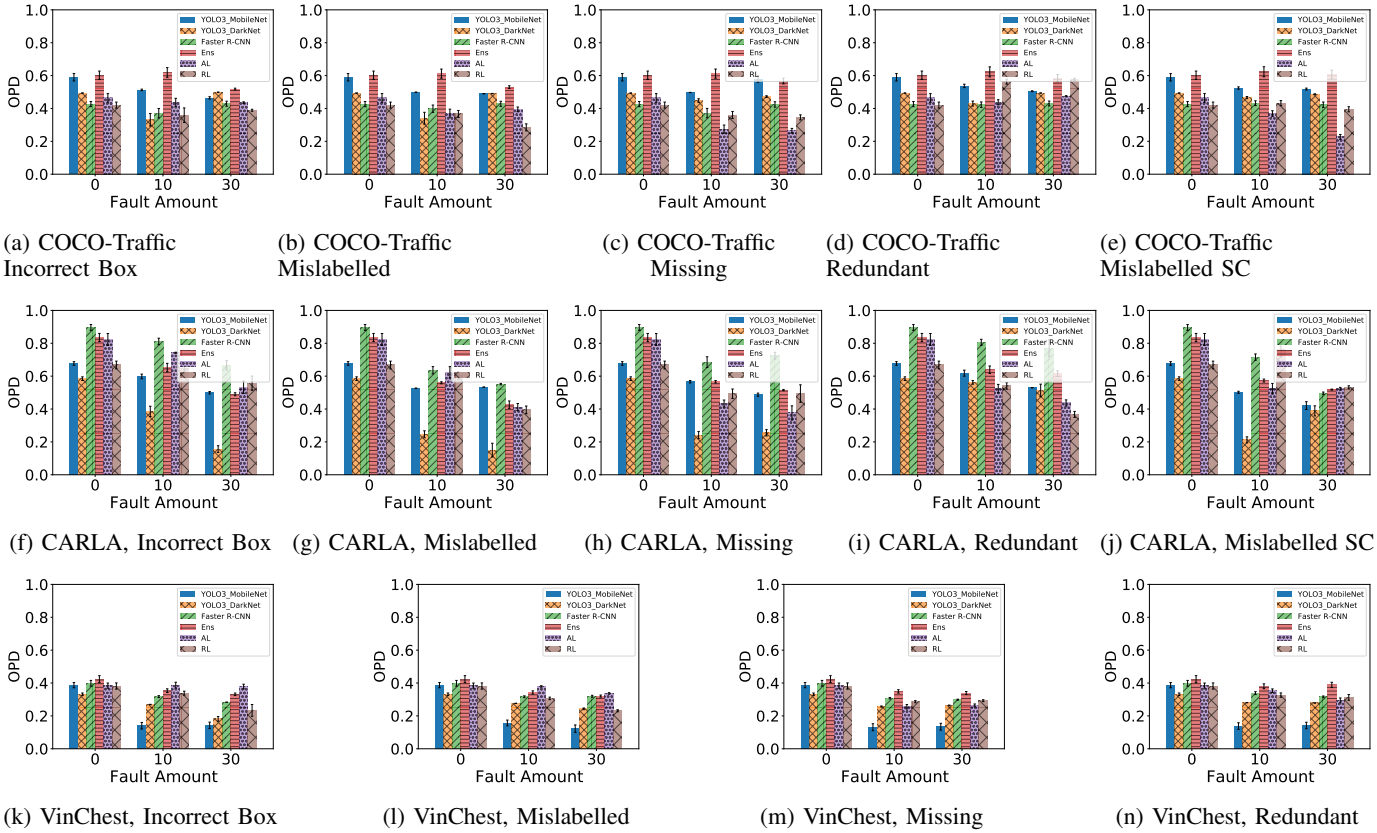


Fig. 4: mAP when a single fault of each fault type is injected in each image of the COCO-Traffic dataset. The error bars indicate the 95% confidence intervals. Higher values are better.

We inject each of the five fault types into 10% and 30% respectively of the total objects in each test dataset and measure the OPD for each configuration. We observe that the OPD is the highest for Faster R-CNN, across all configurations on CARLA and VinChest (Figs. 5f to 5n) except for COCO-Traffic (Figs. 5a to 5e), where YOLOv3 MobileNet outperforms it. This is because two-stage detectors perform worse on images with more objects (Section IV-B). We find that single-stage detectors are more robust in scenes containing higher number of objects (*i.e.*, COCO-Traffic, VinChest) than multi-stage detectors. Multi-stage detector outperform single-stage detectors in CARLA. The average number of objects per image in the test datasets for COCO-Traffic, VinChest, and CARLA are 7.2, 5.4 and 1.4 respectively.



(a) COCO-Traffic Incorrect Box (b) COCO-Traffic Mislabelled (c) COCO-Traffic Missing (d) COCO-Traffic Redundant (e) COCO-Traffic Mislabelled SC

(f) CARLA, Incorrect Box (g) CARLA, Mislabelled (h) CARLA, Missing (i) CARLA, Redundant (j) CARLA, Mislabelled SC

(k) VinChest, Incorrect Box (l) VinChest, Mislabelled (m) VinChest, Missing (n) VinChest, Redundant

Fig. 5: OPD of three different models, along with different protection techniques for each dataset and fault type. The error bars in the results indicate the 95% confidence intervals. Higher values are better.

For each experimental configuration, we also measure the mAP of each model, and compare it to our earlier OPD results. We hypothesize that models, trained with faulty data, with high mAP should also have high OPD. When comparing the mAP across configurations in (Figs. 6f to 6i and 6k to 6n), we make a similar observation, where Faster R-CNN has a higher mAP compared to other models except on COCO-Traffic.

Surprisingly, the difference in OPD between YOLOv3 MobileNet and YOLOv3 DarkNet is much greater than their difference in mAP, with the former having higher OPD. This rejects our earlier hypothesis that models with high mAP also have a high OPD. This indicates YOLOv3 MobileNet is more robust than YOLOv3 DarkNet, especially with superclasses. The opposite, where YOLOv3 DarkNet is more robust than YOLOv3 MobileNet, is observed on VinChest. Because both models use YOLOv3 as their heads, the backbone network appears to affect robustness. For example, DarkNet has more layers than MobileNet, and is more robust on VinChest where X-ray images were higher resolution than traffic images.

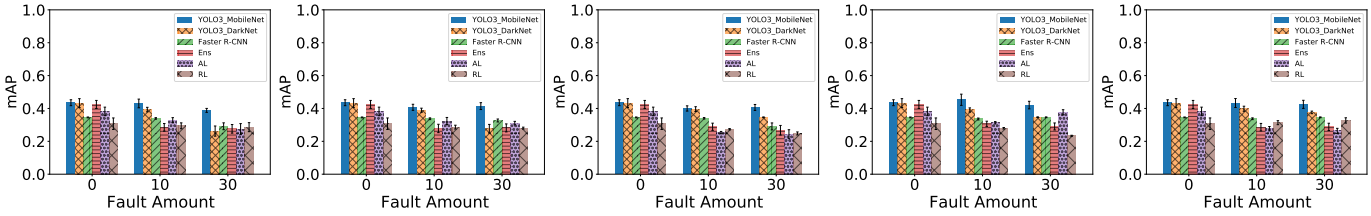
Observation 1 *Single-stage detectors are more robust than multi-stage detectors, in scenes containing more objects.*

D. RQ2: Annotation Fault Types

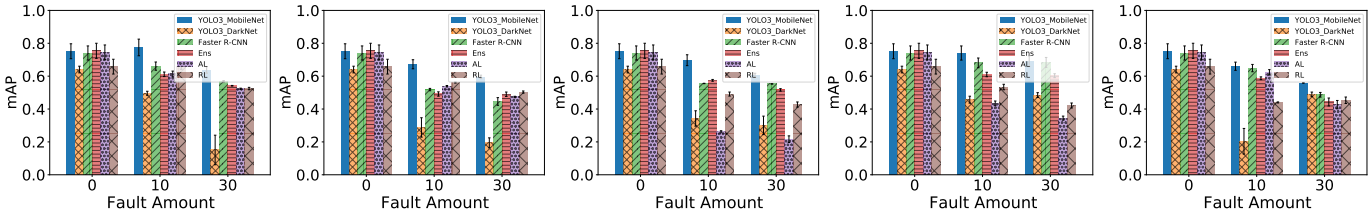
Next, we analyze the results according to fault types. Generally, we observe that as the fault amount increases, OPD decreases, as expected. This trend holds across all fault types.

We also observe that OD models have a lower OPD when incorrect box faults are injected to the training data, but have a similar mAP. This is seen in the mAP values of COCO-Traffic and CARLA under incorrect box, Figs. 6a and 6f in contrast to the OPD values in Figs. 5a and 5f. For instance, the golden mAP of COCO-Traffic for YOLOv3 MobileNet is 0.44, while the mAP for a model with 10% incorrect box is 0.43, which is within the margin of error. However, the golden OPD for COCO-Traffic for YOLOv3 MobileNet is 0.59, while the OPD for the model with 10% incorrect box faults is 0.51. Similar trends are seen for other OD models under this configuration.

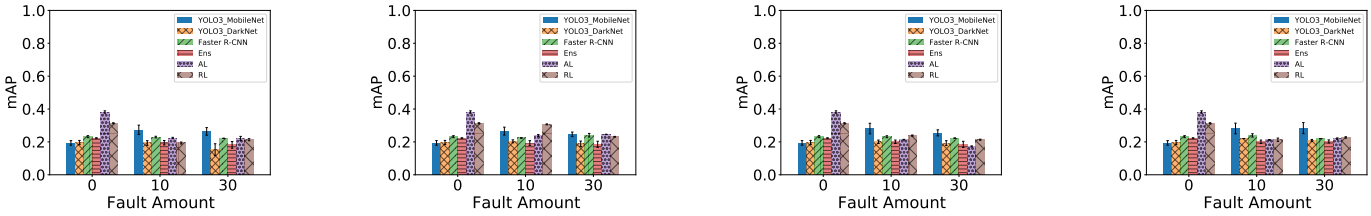
This observation indicates that incorrect box faults cause more predictions belonging to an incorrect superclass (*i.e.* superclass misprediction). We had expected that incorrect box would affect all the object classes equally, as we perturbed the bounding box sizes. However, it appears that smaller objects such as “person” or “stop signs” are more likely to be affected by superclass mispredictions due to incorrect boxes in the training data. While smaller objects are generally more susceptible to mispredictions than larger objects due to their reduced feature space, they are also more prone to superclass mispredictions. Objects of the same superclass usually have a similar bounding box size. For instance, traffic lights, a superclass in COCO-Traffic, have a similar bounding box size,



(a) COCO-Traffic, Incorrect Box (b) COCO-Traffic, Mislabelled (c) COCO-Traffic, Missing (d) COCO-Traffic, Redundant (e) COCO-Traffic, Mislabelled SC



(f) CARLA, Incorrect Box (g) CARLA, Mislabelled (h) CARLA, Missing (i) CARLA, Redundant (j) CARLA, Mislabelled SC



(k) VinChest, Incorrect Box (l) VinChest, Mislabelled (m) VinChest, Missing (n) VinChest, Redundant

Fig. 6: mAP of three different models, along with different protection techniques for each dataset and fault type. The error bars in the results indicate the 95% confidence intervals. Higher values are better.

no matter the type of traffic light. However, when objects are incorrectly sized in the training dataset, the model may misclassify an object into an incorrect superclass (*i.e.* a traffic light to a vehicle) due to incorrect bounding box coordinates.

Compared to other fault types, redundant bounding boxes had the least impact on both mAP and OPD. We can see the effect on mAP (Figs. 6d, 6i and 6n) and OPD (Figs. 5d, 5i and 5n) respectively across datasets. While redundant bounding boxes slightly degraded the localization ability of OD models, it had little impact on their classification ability. Many objects' annotations were still predicted within the IoU threshold, and belonged to the same class as their golden and ground truth categories. As a result, true positive predictions remained high, leading to relatively high mAP and OPD for redundant bounding boxes, compared to other fault types.

Observation 2 *OPD captures the effect of mispredicted superclasses in faulty models, unlike mAP.*

Observation 3 *All fault types have a significant impact on robustness except for redundant bounding boxes.*

E. RQ3: Robust OD Techniques

Finally, we apply the three robust OD techniques to each configuration and compare their OPD values. *We observe that ensembles are the most robust across configurations, followed by AL and RL.* Ensembles have equal or greater OPD than AL and RL in most configurations. However, ensembles have lower OPD than an individual model (Faster R-CNN) in CARLA, seen in Figs. 5f to 5j. Both the single-stage detectors in the ensemble, YOLOv3 MobileNet and YOLOv3 DarkNet, have a significantly lower OPD than Faster R-CNN. Overall, ensembles are more robust as they are the only technique to combine both single and two-stage detectors.

We observe that RL is a more effective technique when deployed at higher fault amounts. This is because Focal Loss (FL) deployed in RL, appears to underperform at low fault amounts. In comparison, ensembles and AL both have high golden OPD, across all datasets. Further, while RL is especially effective for mislabelled superclass, it is also robust to missing annotations, as shown in Figs. 5c, 5h and 5m. FL assigns a lower weight to well-classified training examples, and a higher weight to ambiguous ones [32]. With missing annotations, RL is able to better avoid mispredictions by reducing FPs, though this is at the cost of also reducing TPs.

We also observe that AL is robust in general to the mislabelled class, mislabelled superclass and incorrect box faults, with a relatively high OPD. However, AL is not effective against redundant annotations and missing annotations. This is because AL effectively selects its own subset of training data based on its own feature selection. While this helps for

certain fault types, AL struggles against redundant and missing annotations as there are fewer good examples to learn from.

Observation 4 *Ensembles have the highest overall robustness compared to other techniques. Both AL and RL are robust against some fault types, but RL has a lower golden OPD.*

V. Discussion

Impact of OD Architectures While we find that ensembles are the most robust compared to individual models and other techniques, they are also the most resource intensive, both in terms of memory and runtime, as they require training multiple OD models. During inference, there is also additional overhead to run multiple models in parallel and vote on predictions. In safety-critical applications like AVs, where runtime efficiency is important (especially during inference), practitioners could consider selectively ensembling parts of the OD network. Instead of an ensemble of OD models, one can ensemble only the backbone networks and use a single head. Ensembles of backbone networks are equivalent to ensembles of image classification models, which can benefit from significant optimizations [55]. Alternatively, a single backbone network can be utilized, and the result can be fed into multiple heads (*i.e.* YOLOv3, RPN) and predictions combined through voting.

Threats to Validity We identify two external threats to validity. First, we evaluated robust OD techniques on two-stage detectors only. While this may preclude insights into the techniques on single-stage detectors, we find more existing robust OD techniques [8, 16] are implemented on two-stage versus single-stage detectors due to higher baseline mAP. Second, our evaluation was performed separately against each annotation fault type to help analyze their effects. However, multiple fault types often occur together in OD datasets - considering them together is an avenue for future work.

VI. Related Work

Faults in OD Existing work on evaluating faults in OD models have focused on salt-and-pepper noise and Gaussian blurs in the test data [56, 57]. These types of noise simulate image quality degradation through software defects (*e.g.*, lossy compression) or real-world factors (*e.g.*, low-visibility weather conditions), by inducing pixel corruptions in training images, which soften distinguishing features between object classes. Unlike salt-and-pepper noise, annotation faults do not soften features. For example, bounding boxes associated with incorrect classes can be drawn over any unaltered image. Models will attempt to learn features associated with an incorrect annotation. Annotation faults lead to an abundance of incorrect features [58], while salt-and-pepper noise reduce the number of useful features overall [56], making annotation faults more difficult to tolerate.

Ponce et al. [59] identified gaps in data quality in OD datasets, forming the basis of annotation faults. Subsequent work [53] focused on identifying faulty images, and purging them from datasets. Unlike image classification, where there is exactly one class per image, cleaning tools for OD must handle dense, overlapping objects in images, which is more complicated and error-prone [60]. Furthermore, data cleaning reduces the amount of trainable annotations in data-scarce applications. We focus instead on robust OD techniques that tolerate the presence of faulty training images, without discarding images.

Evaluation Metrics for OD The most popular metric to evaluate OD models is mAP, whose limitations are summarized in Section II. Ceccarelli et al. [61] address the criticality difference between different object classes by assigning a criticality score to each object in relation to other objects in an image, by leveraging sequential trajectory information in a moving scene. In contrast, we focus on a metric that applies to static images without trajectory information. This allows us to more generally evaluate OD accuracy on datasets without sequential context – most OD datasets do not provide this. Liao et al. [62] partially address the distance between critical objects by proposing a distance ratio based on the 2D projection of 3D objects in an image. In contrast, we leverage the use of the commonly-used and well-defined IoU metric between overlapping images, for generality of computation.

Hierarchical Classification Our robustness metric of OPD builds on the idea of *hierarchical classification* (HC), where similar classes are grouped under metaclasses. HC is a well-studied problem where existing work focuses on identifying correlations between classes and perturbing loss functions to work with HC [63, 64]. In particular, Salay et al. [64] find that HC is suited for use in safety-critical systems like AVs. Unlike their work, we do not derive our own class hierarchy - we use superclasses, as established by the dataset authors.

VII. Conclusions

Object detection (OD) datasets used for safety-critical applications inevitably contain annotation faults, which can degrade the ability of models to learn and make correct predictions. Thus, it is crucial to mitigate the effects of such faults on OD.

We systematically study the effects of annotation faults on OD in three steps. First, we develop ODFI, which injects annotation faults into OD datasets. Second, we introduce a new robustness metric OPD that better captures superclass mispredictions by OD models and evaluates the robustness of individual OD models across datasets and fault types. Finally, we compare three representative robust OD techniques.

Our experimental results on three OD datasets show that single-stage detectors trained with faulty annotations perform better in scenes containing more objects, redundant bounding boxes have the least impact on robustness, and ensembles have the highest overall robustness among the robust OD techniques considered. Thus, ensembles are the most effective way to mitigate the impact of annotation faults on OD applications.

Acknowledgements

This work was funded in part by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC), a research gift from Intel, and a Four Year Fellowship from UBC. We thank the Digital Research Alliance of Canada for providing computational resources. We thank the anonymous reviewers of ISSRE'23 for their helpful comments.

REFERENCES

[1] S. S. Banerjee, S. Jha, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "Hands Off the Wheel in Autonomous Vehicles?: A Systems Perspective on over a Million Miles of Field Data," in *Proc. of DSN'18*, 2018.

[2] J. G. Richens, C. M. Lee, and S. Johri, "Improving the accuracy of medical diagnosis with causal machine learning," *Nature Communications*, vol. 11, no. 1, p. 3923, 2020.

[3] Q. Liu, X. Li, S. Yuan, and Z. Li, "Decision-Making Technology for Autonomous Vehicles Learning-Based Methods, Applications and Future Outlook," 2021, arXiv:2107.01110.

[4] T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, and U. Rajendra Acharya, "Automated detection of COVID-19 cases using deep neural networks with X-ray images," *Computers in Biology and Medicine*, vol. 121, p. 103792, 2020.

[5] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object Detection With Deep Learning: A Review," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[6] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," 2014, arXiv:1405.0312.

[7] S. Enderes, "The impact of annotation errors on neural networks," https://understand.ai/blog/annotation/machine-learning/autonomous-driving/2021/06/01/impact-of-annotation-errors-on-neural-networks.html, accessed: 2023-03-06.

[8] C. Liu, K. Wang, H. Lu, Z. Cao, and Z. Zhang, "Robust Object Detection With Inaccurate Bounding Boxes," in *Proc. of ECCV'22*, 2022.

[9] A. Chan, A. Gujarati, K. Pattabiraman, and S. Gopalakrishnan, "The Fault in Our Data Stars: Studying Mitigation Techniques against Faulty Training Data in Machine Learning Applications," in *Proc. of DSN'22*, 2022.

[10] P. Koopman and M. Wagner, "Challenges in Autonomous Vehicle Testing and Validation," *SAE International Journal of Transportation Safety*, 2016.

[11] E. Medvedeva, "Moving Object Detection in Noisy Images," in *Proc. of MECO'19*, 2019.

[12] S. Cygert and A. Czyżewski, "Toward Robust Pedestrian Detection With Data Augmentation," *IEEE Access*, 2020.

[13] C. Li *et al.*, "Object Detection based on OcSaFPN in Aerial Images with Noise," 2020, arXiv:2012.09859.

[14] T. Sharma *et al.*, "Deep Learning-Based Object Detection and Scene Perception under Bad Weather Conditions," *Electronics*, 2022.

[15] S. Chadwick and P. Newman, "Training Object Detectors With Noisy Data," in *Proc. of IV'19*, 2019.

[16] X. Liu, W. Li, Q. Yang, B. Li, and Y. Yuan, "Towards Robust Adaptive Object Detection under Noisy Annotations," in *Proc. of CVPR'22*, 2022.

[17] A. K. Heger, L. B. Marquis, M. Vorvoreanu, H. Wallach, and J. Wortman Vaughan, "Understanding Machine Learning Practitioners' Data Documentation Perceptions, Needs, Challenges, and Desiderata," *Proceedings of the ACM on Human-Computer Interaction*, 2022.

[18] C. Consortium, "Data format," https://cocodataset.org/#format-data, accessed: 2023-05-01.

[19] J. Hui, "mAP (mean Average Precision) for Object Detection," 2018.

[20] A. F. Gad, "Evaluating Object Detection Models Using Mean Average Precision (mAP)," *PaperspaceBlog*, 2020.

[21] D. Shah, "Mean average precision (MAP) explained: Everything you need to know," 2022.

[22] M. Everingham *et al.*, "The PASCAL Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision*, 2015.

[23] H. Rezatofighi *et al.*, "Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression," in *Proc. of CVPR'19*, 2019.

[24] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *Proc. of IWSSIP'20*, 2020.

[25] A. Chan, N. Narayananan, A. Gujarati, K. Pattabiraman, and S. Gopalakrishnan, "Understanding the Resilience of Neural Network Ensembles against Faulty Training Data," in *Proc. of QRS'21*, 2021.

[26] D. Kirchhoff and P. Hoang, "COCO Traffic," https://github.com/daved01/cocoTraffic, 2021.

[27] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018, arXiv:1804.02767.

[28] A. Casado-García and J. Heras, "Ensemble Methods for Object Detection," in *Proc. of ECAI'20*, 2020.

[29] J. Gao, J. Wang, S. Dai, L.-J. Li, and R. Nevatia, "NOTE-RCNN: NOise Tolerant Ensemble RCNN for Semi-Supervised Object Detection," in *Proc. of ICCV'19*, 2019.

[30] P. Wei, J. E. Ball, and D. T. Anderson, "Fusion of an Ensemble of Augmented Image Detectors for Robust Object Detection," *Sensors*, 2018.

[31] S. Paul, S. Chandrasekaran, B. S. Manjunath, and A. K. Roy-Chowdhury, "Exploiting Context for Robustness to Label Noise in Active Learning," 2020, arXiv:2010.09066.

[32] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[33] Z. Hu, K. Gao, X. Zhang, and Z. Dou, "Noise Resistant Focal Loss for Object Detection," in *Proc. of PRCV'20*, 2020.

[34] R. Rosales, P. Munoz, and M. Paulitsch, "Exploring Resiliency to Natural Image Corruptions in Deep Learning using Design Diversity," 2023, arXiv:2303.09283.

[35] K.-H. Chow and L. Liu, "Boosting Object Detection Ensembles with Error Diversity," in *Proc. of ICDM'22*, 2022.

[36] P. Soviany and R. T. Ionescu, "Optimizing the Trade-off between Single-Stage and Two-Stage Object Detectors using Image Difficulty Prediction," 2018, arXiv:1803.08707.

[37] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2013.

[38] F. C. Akyon, S. O. Altinuc, and A. Temizel, "Slicing Aided Hyper Inference and Fine-tuning for Small Object Detection," in *Proc. of ICIP'22*, 2022.

[39] A. Hantson, "CARLA Dataset," https://universe.roboflow.com/alec-hantson-student-howest-be/carla-izloa, 2022, accessed: 2023-05-02.

[40] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Proc. of CoRL'17*, 2017.

[41] H. Q. Nguyen *et al.*, "VinDr-CXR: An open dataset of chest X-rays with radiologist's annotations," 2020, arXiv:2012.15029.

[42] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Proc. of NIPS'15*, 2015.

[43] S. Bouraya and A. Belangour, "Deep Learning based Neck Models for Object Detection: A Review and a Benchmarking Study," *International Journal of Advanced Computer Science and Applications*, 2021.

[44] C. Vasconcelos, V. N. Birodkar, and V. Dumoulin, "Proper Reuse of Image Classification Features Improves Object Detection," in *Proc. of CVPR'22*, 2022.

[45] A. Sarda, S. Dixit, and A. Bhan, "Object Detection for Autonomous Driving using YOLO algorithm," in *Proc. of ICIEM'21*, 2021.

[46] D. Mijić, M. Brisinello, M. Vranješ, and R. Grbić, "Traffic Sign Detection Using YOLOv3," in *Proc. of ICCE-Berlin'20*, 2020.

[47] U. Nepal and H. Eslamiat, "Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty

UAVs," *Sensors*, 2022.
[48] T. Rahmat, A. Ismail, and S. Aliman, "Chest x-ray image classification using faster r-cnn," *Malaysian Journal of Computing*, 2019.
[49] Y. Xiao *et al.*, "A Cascade and Heterogeneous Neural Network for CT Pulmonary Nodule Detection and Its Evaluation on both Phantom and Patient Data," *Computerized Medical Imaging and Graphics*, 2021.
[50] T. Bu, X. Zhang, C. Mertz, and J. M. Dolan, "CARLA Simulated Data for Rare Road Object Detection," in *Proc. of ITSC'21*, 2021.
[51] I. Panshin and S. Zlobin, "2nd-place-solution-for-VinBigData-Chest-X-ray-Abnormalities-Detection," https://www.kaggle.com/c/vinbigdata-chest-xray-abnormalities-detection/discussion/229696, accessed: 2023-05-31.
[52] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, "Bounding Box Regression with Uncertainty for Accurate Object Detection," 2019, arXiv:1809.08545.
[53] J. Ma, Y. Ushiku, and M. Sagara, "The Effect of Improving Annotation Quality on Object Detection Datasets: A Preliminary Study," in *Proc. of CVPRW'22*, 2022.
[54] J. Murdoch, "How I found nearly 300,000 errors in MS COCO," https://medium.com/@jamie_34747/how-i-found-nearly-300-000-errors-in-ms-coco-79d382edf22b, accessed: 2022-11-28.
[55] S. Liu *et al.*, "Deep Ensembling with No Overhead for either Training or Testing: The All-Round Blessings of Dynamic Sparsity," in *Proc. of ICLR'22*, 2022.
[56] Z. Pezzementi, T. Tabor, S. Yim, J. Chang, B. Drozd, D. Guttendorf, M. Wagner, and P. Koopman, "Putting Image Manipulations in Context: Robustness Testing for Safe Perception," in *Proc. of SSRR'18*, 2018.
[57] S. Dodge and L. Karam, "Understanding How Image Quality Affects Deep Neural Networks," in *Proc. of QoMEX'16*, 2016.
[58] L. Zhao, "Active Learning With Unreliable Annotations," Ph.D. dissertation, 2013.
[59] J. Ponce *et al.*, *Dataset Issues in Object Recognition*, 2006.
[60] A. Peskin, B. Wilthan, and M. Majurski, "Detection of Dense, Overlapping, Geometric Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
[61] A. Ceccarelli and L. Montecchi, "Evaluating the Consequences of Object (mis)Detection from a Safety and Reliability Perspective: Discussion and Measures," 2022, arXiv:2203.02205.
[62] H.-C. Liao, C.-H. Cheng, H. Esen, and A. Knoll, "Improving the Safety of 3D Object Detectors in Autonomous Driving using IoGT and Distance Measures," 2023, arXiv:2209.10368.
[63] C. N. Silla and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, 2011.
[64] R. Salay, M. Angus, and K. Czarnecki, "A Safety Analysis Method for Perceptual Components in Automated Driving," in *Proc. of ISSRE'19*, 2019.