Evaluating the Consequences of Object (mis)Detection from a Safety and Reliability Perspective: Discussion and Measures

Andrea Ceccarelli,[1] **Leonardo Montecchi**[2]

[1]University of Florence, Florence, Italy
[2]Norwegian University of Science and Technology, Trondheim, Norway

Abstract

We argue that object detectors in the safety critical domain should prioritize detection of objects that are most likely to interfere with the actions of the autonomous actor. Especially, this applies to objects that can impact the actor's safety and reliability. To quantify the impact of object (mis)detection on safety and reliability in the context of autonomous driving, we propose new object detection measures that reward the correct identification of objects that are most dangerous and most likely to affect driving decisions. To achieve this, we build an object criticality model to reward the detection of the objects based on proximity, orientation, and relative velocity with respect to the subject vehicle. Then, we apply our model on the recent autonomous driving dataset nuScenes, and we compare nine object detectors. Results show that, in several settings, object detectors that perform best according to the nuScenes ranking are not the preferable ones when the focus is shifted on safety and reliability.

1 Introduction

Object detection is fundamental in emerging safety-critical applications, and in particular it is a major pillar of autonomous driving applications. Object detectors are evaluated against datasets using widely acknowledged measures (Powers 2020) and well-defined routines (Geiger et al. 2013), (Everingham et al. 2015), (Cordts et al. 2016), (Caesar et al. 2020), (Houston et al. 2021), allowing a fair comparison. In particular, the Average Precision, first presented in (Salton and McGill 1983), is currently deemed the most suitable approach to summarize and rank the performance of object detectors.

However, we argue that current measures for object detection do not match the demands and peculiarities of autonomous vehicles and safety-critical systems in general, i.e., systems whose failure may lead to harmful consequences (Avizienis et al. 2004). Evaluations based on Average Precision typically judge how well a detector detects objects, without discriminating based on the current behavior of these objects, or on their possibility to interfere with the vehicle. In other words, the relevance for the driving task is not considered. To clarify, let us consider the typical pipeline for autonomous driving (Grigorescu et al. 2020): the subject vehicle is sensing the surroundings to perform object detection, and the detection output is used for path planning. Let us now consider two other vehicles in the sensed scenario, one directed straight towards the subject vehicle, in a colliding trajectory, and one headed away from the subject vehicle. Clearly, for the safety of the driving task, it is critical to detect the first one, while detection of the second vehicle is not relevant at all. Unfortunately, this is not captured by the measures currently used in object detection, which consider both objects as equally relevant.

In this paper we elaborate on how to measure performance of object detectors in the safety-critical domain, with specific contextualization to the domain of autonomous driving, and we identify the need of an object criticality model and related measures. As key requirement, the desired measures should reward the detection of those objects that may interfere with the subject vehicle, and that are relevant for the safe and reliable execution of the driving task. Also, to be practically useful, the proposed measures have to be in a defined range, and be summarized by an overarching unifying measure.

Consequently, we build an object criticality model that assigns a criticality score to each object, based on ground truth and estimated object distance, colliding trajectory, and time to collisions. Such criticality scores contribute to compute measures, named reliability-weighted precision and safety-weighted recall, that weight correct object detections and misdetections based on the impact on safety and reliability of the driving task. Last, a summarizing measure, named Critical Average Precision, allows ranking detectors according to such safety- and reliability-oriented measures.

The object criticality model and the related measures are exercised on the nuScenes dataset, with nine 3D-object detectors. We show that, under numerous settings, the ranking we obtain differs from the one achieved using the nuScenes evaluation library, which relies on traditional measures. Amongst implications, this result questions the usual approach to rate and select the most suitable object detector for the autonomous driving domain.

The rest of the paper is organized as follows. Section 2 presents basic notions and the related works. Section 3 shows the object criticality model and the measures we are introducing. Section 4 describes the experiments based on nine object detectors and the nuScenes dataset. Section 5 illustrates the

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

results, in which the object detectors are ranked according to our and traditional measures, and differences are discussed. Section 6 concludes the paper.

2 Background and Related Works

2.1 Object detection and its evaluation

We report the minimal set of notions on object detection that we require to present the choices made in our work, to make the document self-contained.

To describe the spatial location and extent of a detectable object, in this paper for simplicity we only consider bounding boxes, although alternative approaches, e.g., (Liu et al. 2020), are applicable to our object criticality model as well.

Object detectors compute bounding boxes with an assigned confidence score. Then, a *detection threshold* is applied as a configuration parameter: all bounding boxes with confidence score above the detection threshold are predictions. The classification of true positives (TPs), false positives (FPs), and false negatives (FNs), is based on some definition of distance between the predicted bounding boxes and the ground truth bounding boxes. In this paper, we use the distance between their center points (Caesar et al. 2020): a detected object is considered a TP if the distance between the ground truth bounding box and the detected bounding box is closer than a *distance limit*.

The conventional approach to the evaluation of object detectors consists of measures that are derived from the count of TP, FP, and FN. These form the basis for our object criticality model defined in Section 3, and they are briefly reviewed here (Powers 2020). *Precision*, $P = TP/(TP + FP)$, indicates how many of the selected items are relevant. *Recall*, $R = TP/(TP + FN)$, indicates how many of the existing relevant items are selected. The *Precision-Recall Curve* shows the tradeoff between precision and recall for different detection thresholds. Currently, the most frequently used summarizing measure is *Average Precision* (AP) (Everingham et al. 2010), which summarizes the precision-recall curve as the weighted mean of precision scores achieved at different detection thresholds, using the increase in recall from the previous detection threshold as the weight. More precisely, $AP = \sum_n (R_n - R_{n-1})\, P_n$, where P_n and R_n are the precision and recall at the n-th detection threshold. In this paper, in agreement with (Caesar et al. 2020), we calculate AP only for recall and precision above or equal to 0.1: we remove cases in which recall or precision is less than 0.1 in order to minimize the impact of noise commonly seen in regions with low precision or low recall.

2.2 Safety of the object detection task

A *safety*-critical (computer) system is one whose malfunction could lead to unacceptable consequences, like harm to users or to the environment. A typical example is an autonomous vehicle, whose malfunction (of whatever cause) may lead to a collision. Instead, *reliability* describes the continuity of correct service, which can be temporarily disrupted, for example, to avoid situations that are potentially dangerous (Avizienis et al. 2004).

Considering object detectors, some incorrect predictions may lead to catastrophic consequences, and therefore have the maximum impact on safety, while others may have irrelevant impact. Further, some false positives may cause an unnecessary interruption of the service, and therefore they impact the reliability. However, to evaluate object detectors, measures from Section 2.1 are typically used, without considering the different impact of each detection mistake. This also applies to the wide domain of autonomous driving, and it becomes evident when considering the measures used in object detection challenges for autonomous driving. For example, in challenges for KITTI (Geiger et al. 2013), CityScapes (Cordts et al. 2016), Waymo (Sun et al. 2020), or nuScenes (Caesar et al. 2020), evaluation measures revolve around Average Precision and the concepts summarized in Section 2.1.

Up to now, very few approaches have attempted to define safety or reliability measures for object detectors; to the best of our knowledge, the only works which targets a similar goal are focusing on safety, and are (Topan et al. 2022), (Bansal et al. 2021), (Lyssenko et al. 2021), (Volk et al. 2020). Noteworthy, they all appeared in the last two years, which underlines a recent understanding of the relevance of the subject, and they are all in the autonomous driving domain. The work in (Bansal et al. 2021) ranks each object in three categories (imminent collision, potential collision, no collision), based on its collision risk. Instead, in (Topan et al. 2022) the authors define critical zones in which accurate perception is mandatory. On a similar position, the authors of (Lyssenko et al. 2021) argue the relevance of identifying a distance up to which all pedestrian are detected. The closest approach to our work is (Volk et al. 2020), where the authors combines scores measuring detection quality, collision potential, and time needed to make the detection, to compute a safety score of a test scenario, in 5 classes from insufficient to excellent.

With respect to the reviewed works, the object criticality model we propose includes both safety and reliability issues of the driving task. This is important, because safety by itself (detect everything which is potentially dangerous) can be enforced by low precision and high recall, i.e., low false negatives at the cost of many false positives. Instead, balancing both reliability and safety issues, our model provides a standalone evaluation of object detectors; we experimentally show that, when safety and reliability are central for the selection of the object detector, the most suitable one may not be the best performing according to traditional measures.

3 Object Criticality Model

Our model is based on assigning a *criticality* value to each object in the scene, and then computing object detection measures that consider this criticality. The description of the model is independent of the sensors used to capture the scene (e.g., cameras or lidars) and of the type of objects.

3.1 Requirements and assumptions

The application of the model requires i) a subject vehicle (named *ego* afterwards) that captures the scene with sensors as cameras and lidars, and ii) objects (other vehicles, pedestrians, etc.) that are within line-of-sight to ego and that are

consequently captured by the sensors. This is the very typical situation of an autonomous vehicle that performs object detection.

We assume that the following ground truth information is available: i) 3D bounding boxes describing the size of the objects; ii) coordinates of ego and of the objects; iii) velocity of ego and of the objects. The most recent automotive datasets have very rich meta-data, typically including the above information; for example in Section 4 and Section 5 we will use nuScenes (Caesar et al. 2020), which satisfies our assumptions. Clearly, the ground truth is required only to evaluate the model, and not in case of model operation.

Further, we assume that the object detector produces as output: i) the computed 3D bounding boxes, ii) the estimated distance of detected objects from ego, and iii) the estimated velocity of objects. In other words, the model for object detection conflates detection, tracking, and dynamics: this is done in several 3D object detectors, which include the above estimates in their output. Noteworthy, these estimates are computed in the object detection challenges of the nuScenes community, which will be our reference for the experiments in Section 4 and Section 5.

For simplicity, when computing coordinates of objects and their distance from ego, we consider only the (x, y) coordinates, i.e., we ignore the vertical dimension. For the nuScenes dataset we use in this paper this is not an issue, because data was collected on mostly flat lands. Extending the model to consider the elevation is definitely possible, only at the cost of slightly more complex geometric computations.

3.2 Structure of the model

We call ego the roving vehicle that mounts the sensors and collects data from the environment, and we call object B any other object. Note that for ego we only have ground truth values, i.e., the object detector does not predict its own velocity or position.

The construction of our model is organized in 3 steps, which are repeated for each object B within the line of sight of ego, and for both the ground truth values and the predicted values of B.

The first step (Section 3.3) is the analysis of the collision scenario involving B and ego. In this step we calculate indicators that will be later used to define the criticality of B. In particular, we calculate i) the initial distance d between ego and B, ii) the closest distance r that ego and B would reach, and iii) the time Δt that ego and B require to reach such distance. These values are input to the following step, together with the current position and velocity of ego and B.

The second step (Section 3.4) is the calculation of *criticality weights* that are assigned to each object B. These are κ_d, κ_r, and κ_t, and they are based, respectively, on the three values calculated in the first step. These weights indicate the relevance of B for the driving task: weights are higher if it is more likely that B may affect the behavior of ego. Such weights are used as rewards or penalties depending, respectively, on whether the object has been detected or missed.

The third step (Section 3.5) exploits the assigned criticality to construct aggregate *safety and reliability measures* that allow comparing different detectors.

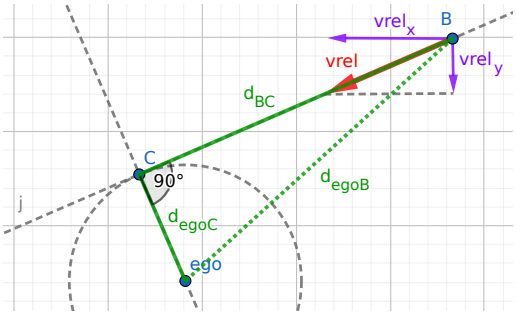

Figure 1: Geometrical representation of the main elements of our model.

3.3 Analytical characterization of the collision scenario

We refer to Figure 1 for a visual representation of the collision scenario analyzed in this section.

We define $ego = (ego_x, ego_y)$ the position of ego, and $B = (B_x, B_y)$ the position of the object B in the captured scene. Further, we define, in vector form, $v_{ego} = (v_{ego_x}, v_{ego_y})$ the velocity of ego, and $v_B = (v_{B_x}, v_{B_y})$ the velocity of B. We compute the relative velocity of B with respect to ego, as $v_{rel} = (v_{rel_x}, v_{rel_y}) = (v_{B_x} - v_{ego_x}, v_{B_y} - v_{ego_y})$, that is, the vectorial difference of the velocity of ego and the velocity of B. This allows simplifying the subsequent calculations: we can consider ego as stationary, while B is moving with the velocity resulting from the difference of the two velocity vectors v_{ego} and v_B.

Then, we identify the shortest distance from ego at which object B will pass if both continue moving with the same velocity. This is the distance between ego and point $C = (C_x, C_y)$, with C being the point closest to ego on the trajectory of B. Point C can also be thought as the tangent point between the line representing the trajectory of B and a circle centered on ego.

Point $C = (C_x, C_y)$ can be computed as the intersection of two lines, using basic Euclidean geometry. The line defining the direction of the relative movement of B is obtained from the general equation of a line, i.e., $y - y_0 = m(x - x_0)$. We are looking for the line passing from point (B_x, B_y) and whose angular coefficient (i.e., orientation with respect to the x axis) is given by the ratio between the y and x components of the relative velocity v_{rel} (refer again to Figure 1).

The shortest distance between such line j and the position of ego lies on the line perpendicular to j passing from ego. By definition, a line perpendicular to one having coefficient m has coefficient $m' = -1/m$. Point $C = (C_x, C_y)$ is then the point at which these two lines intersect, which is obtained by solving Equation 1 below. Computations are omitted for brevity.

$$\begin{aligned} C_y &= \frac{v_{rel_y}}{v_{rel_x}} (C_x - B_x) + B_y, \\ C_y &= -\frac{v_{rel_x}}{v_{rel_y}} (C_x - ego_x) + ego_y \end{aligned} \tag{1}$$

Then, applying the Euclidean distance, we can easily compute: i) the distance d_{egoB} between ego and B, ii) the distance

d_{egoC} between ego and C, and iii) the distance d_{BC} between B and C.

Assuming that both ego and B continue moving with the same velocity, the time Δt that B needs to reach the collision point C is then computed as the distance divided by the scalar speed of B, i.e., $\Delta t = d_{BC}/|v|$, where $|v| = \sqrt{v_x^2 + v_y^2}$. We recall that ego is considered to be stationary, while B moves with a relative velocity obtained as the difference of the velocities of the two objects.

We note that including acceleration would better characterize objects' movement; however, since acceleration is quadratic with respect to space, any estimation error would be greatly amplified, introducing unnecessary noise in the final measures.

Finally, note that the model above exhibits some corner cases, for example, when ego and B are moving at the exact same velocity, Δt is undefined. We treat these rare cases by skipping the model calculation and setting the criticality values to conservative fallback values.

3.4 Computation of criticality weights

The collision scenario above is used to assign criticality to objects. The idea is inspired by reliability analysis (Trivedi 1982), in which metrics like *reliability* (or *safety*) are defined in the interval $[0, 1]$. However, we do not propose probabilities.

Each object B, either identified by the object detector or ground truth, is assigned a *criticality weight* $\kappa(B)$. This weight is obtained combining three criticality values $\kappa_d(B)$, $\kappa_r(B)$, and $\kappa_t(B)$, as explained later. Note that for a given object B, its criticality $\kappa(B)$ may be different if calculated with its predicted properties (e.g., position and velocity) or the ground truth ones. Furthermore, for some objects we may have ground truth values only (FNs) or predicted values only (FPs). When needed, we indicate with $\kappa'(B)$ the criticality weight computed with *predicted* properties of object B, as opposed to $\kappa(B)$ that is calculated based on the ground truth.

Criticality Scores. The *Distance Criticality*, $\kappa_d(B)$, is based on the distance d_{egoB} between ego and the object B. This score does not depend on velocity, but only on the position of objects in the scene. We want the score to be maximum when the distance from ego to B is zero, and then decrease to zero when reaching a maximum distance $D_{max} > 0$.

We compute the weight $\kappa_d(B)$ as a second-degree equation (downward parabola) passing from points $(0, 1)$ and $(D_{max}, 0)$. That is, the maximum value is 1.0 when $d_{egoB} = 0$ and it decreases as d_{egoB} increases, reaching 0 when $d_{ego} = D_{max}$. The parabola shape allows the criticality to decrease non-linearly with respect to the distance: the decrease is slow for values close to zero (i.e., close to the vehicle), and it gets faster when approaching D_{max} (i.e., far from the vehicle). We also need to enforce that $\kappa_d(B)$ is always in the interval $[0, 1]$, and therefore the final equation is:

$$\kappa_d(B) = \max\left(0, -\frac{1}{Z^2}x^2 + 1\right), x = d_{egoB}, Z = D_{max}. \quad (2)$$

The *Collision Distance Criticality*, $\kappa_r(B)$, is based on the distance between ego and the potential collision point C. It is an indicator of how close to ego the object is likely to pass. $\kappa_r(B)$ is calculated using the same rationale of $\kappa_d(B)$ (Equation 2), with $x = d_{egoC}$ and $Z = R_{max}$, where $R_{max} > 0$ is the maximum considered collision distance, beyond which the corresponding criticality is zero.

Similarly, the *Collision Time Criticality*, $\kappa_t(B)$, is based on the time Δt for B to reach the potential collision point. All the other things unchanged, this score depends on the (relative) velocity of the object B with respect to ego. This score is again calculated based on Equation 2, with $x = \Delta t$ and $Z = T_{max}$.

Object Criticality. The final criticality $\kappa(B)$ is obtained by combination of the three criticality scores $\kappa_d(B)$, $\kappa_r(B)$, $\kappa_t(B)$. The resulting measure is defined following four requirements: i) it should range in the interval $[0, 1]$; ii) it should be 0 if all values are zero; iii) it should be 1 if at least one of the values is 1; and iv) it should increase if any of the three values increases.

Inspired again by classic reliability analysis (Trivedi 1982), our final criticality weight is then computed as:

$$\kappa = 1 - (1 - \kappa_d) \cdot (1 - \kappa_r) \cdot (1 - \kappa_t). \quad (3)$$

The final criticality $\kappa(B)$ is therefore a measure of: how much the object is close, how much it is likely to pass close in the near future, and how much time is available to react.

3.5 Safety- and reliability-based measures

We exploit the above criticality scores to remodel the traditional recall and precision measures, such that they are more oriented towards reflecting the safety and the reliability offered by object detectors.

Reliability measures the *continuity of correct service*. For a reliable driving task, a good object detector should *not* predict false positives that correspond to dangerous situations, because they could lead to an interruption of the driving task. For example, false positives may cause an unnecessary brake; instead, the continuity of the driving mission may require to consider some risks of collision as unavoidable. This clearly conflicts with safety (which aims to minimize risks), but it is widely accepted that safety and reliability have different goals (Avizienis et al. 2004) and may be conflicting requirements.

For this reason, we measure the reliability of the detection task through a revised definition of *precision*. The idea is that false positives are penalizing the continuity of the driving process, with a greater impact the closer they are, or are likely to be, to ego. We weight TPs and FPs according to the criticality $\kappa(B)$ of the associated object B. In simpler words, when a non-existing object is detected, we do not add 1 to the count of FPs, but instead we add its criticality; the same applies for TPs.

For a correctly detected object we may use the criticality computed either using the ground truth (κ) or the predicted values (κ'): we use ground truth values at the numerator, and predicted values at the denominator. The idea is that the detector might detect a greater criticality (denominator) than what is actually present (numerator), which reduces reliability

of the driving task. Also, clearly we do not have ground truth values for FPs, because those objects do not exist.

We can then define the *reliability-weighted precision* as:

$$P_{\mathcal{R}} = \frac{\sum_{B \in TP^*} \kappa(B)}{\sum_{B \in TP^*} \kappa'(B) + \sum_{B \in FP^*} \kappa'(B)}, \quad (4)$$

where TP^* is the set of true positive objects, and FP^* is the set of false positive objects. Note that the $P_{\mathcal{R}}$ may in principle raise above 1, in case the detected criticality is significantly lower than the ground truth. To be consistent with the classic definition of precision, we limit the maximum value of $P_{\mathcal{R}}$ to 1.

Safety is instead the *absence of catastrophic consequences*. To ensure safety, the object detector must detect as much as possible of the dangerous objects, even at the cost of raising some false alarms. A safety measure should then reflect how much of the existing criticality has been detected by the object detector. The proposed measure is adapted from the recall, using the ground truth values at the denominator and the detected values at the numerator. Clearly, we do not have predicted values for FNs, which are objects that have been missed. Therefore, we define the *safety-weighted recall* as:

$$R_{\mathcal{S}} = \frac{\sum_{B \in TP^*} \kappa'(B)}{\sum_{B \in TP^*} \kappa(B) + \sum_{B \in FN^*} \kappa(B)} \quad (5)$$

where TP^* is the set of true positive objects and FN^* is the set of false negative objects. Also for $R_{\mathcal{S}}$ we limit its maximum value to 1.

Critical Average Precision The proposed criticality values depend on three parameters, namely D_{max}, R_{max}, and T_{max}. We can compute $P_{\mathcal{R}}$ and $R_{\mathcal{S}}$ for different values of these parameters, to understand their evolution when different subsets of objects are considered. In analogy to the precision-recall curve (see Section 2.1), this allows computing several $P_{\mathcal{R}}$-$R_{\mathcal{S}}$ curves, one for each combination of values $(D_{max}, R_{max}, T_{max})$; consequently, we can compute the Critical Average Precision AP_{crit} from each of the $P_{\mathcal{R}}$-$R_{\mathcal{S}}$ curves, based on our definitions of $P_{\mathcal{R}}$ and $R_{\mathcal{S}}$.

Depending on the driving scenario and the intended system in which the object detector is deployed, different values of D_{max}, R_{max}, and T_{max} may be favored. For example, an object detector which is very good on $P_{\mathcal{R}}$ could be safely used on a highway under low traffic conditions; but if it is not good on $R_{\mathcal{S}}$, it should not be used in an urban scenario, where cars may approach from different directions at essentially any angle.

4 Case Study on the nuScenes Dataset

To exercise our model, we choose the nuScenes dataset for the following reasons: i) it is very recent and extensive, forged with the latest sensor technology; ii) very recent object detectors are available; iii) it includes all the necessary information to apply the model presented in Section 3.

NuScenes (Caesar et al. 2020) is a recent large-scale dataset for autonomous driving that reports scenes collected from a vehicle. The dataset comprises 1000 scenes, each being 20 seconds long and fully annotated with 3D bounding boxes. *Keyframes* are sampled every 0.5 seconds; five intermediate frames are collected between keyframes.

Following common practices in datasets of this kind (Geiger et al. 2013), (Sun et al. 2020), nuScenes defines an object detection task and proposes related measures to officially rank object detectors on its website. The detection task in nuScenes consists in predicting the objects at each keyframe time t, using sensors data collected between $(t - 0.5, t]$ seconds (five intermediate frames). Detectable objects are all objects within 50 meters from ego and with line of sight. For each object, ground truth 3D bounding boxes, attributes (e.g., orientation), and velocities are provided. A detection is successful if the distance between the centers of the predicted and ground-truth bounding boxes is less than a distance limit l; four different values of l are considered, which are $l \in \{0.5, 1, 2, 4\}$ meters. For brevity of the discussion, the only objects we consider are cars.

We select nine 3D object detectorsfrom the model zoo of mmdetection3d (Contributors 2020), an open-source object detection toolbox based on PyTorch for 3D detection. We present the object detectors below; each detector is matched to an acronym to easily distinguish it in the rest of the paper.

FCOS (Wang et al. 2021) and its evolution *PGD* (Wang et al. 2022) use visual cameras only. The backbone is a pretrained ResNet101 with deformable convolutions (Dai et al. 2017). The neck is the Feature Pyramid Network (FPN, (Lin et al. 2017a)), which generates a pyramid of feature maps. The head that produces final predictions (deciding on object class, location, etc.) relies on an approach similar to RetinaNet (Lin et al. 2017b), which applies shared heads to operate detection of multiple targets. PGD head also includes a branch to improve the estimation of distance depth.

The other seven object detectors (see Table 1) process lidar's pointcloud and they are based on the Pointpillars (Lang et al. 2019) network. Pointpillars is well-known both for its speed and its accuracy. It exploits an encoder that learns features on pillars (vertical columns) of the point cloud to predict 3D oriented bounding boxes for objects. The Pointpillars network consists of three main stages: i) a feature encoder

Table 1: The seven lidar-based object detectors in use.

Acronym	Short description
FPN	Backbone is FPN.
REG400	Backbone is the REGNETX-400MF from (Radosavovic et al. 2020).
REGSEC	Similar to REG400, but includes the neck SECOND (Yan, Mao, and Li 2018).
REG1.6	Backbone is the REGNETX-1.6GF model from (Radosavovic et al. 2020).
SEC	Backbone is FPN, neck is SECOND (Yan, Mao, and Li 2018).
SSN	As in SEC above, but it adds the shape-aware grouping heads from (Zhu et al. 2020).
SSNREG	As in SSN, but the backbone is REGNETX-400MF from (Radosavovic et al. 2020).

network that converts a point cloud to a structured representation, namely a sparse pseudoimage; ii) a 2D convolutional backbone to process the pseudo-image into high-level representation, extracting the features map upon which the rest of the network is used; and iii) a detection head that detects and regresses 3D bounding boxes. We consider seven alternatives based on Pointpillars; essentially, they use the pillar-based method from (Lang et al. 2019) to convert the point cloud in a sparse pseudoimage, and differentiate from (Lang et al. 2019) by applying different backbones, and optionally changing the necks and heads.

We ran all the models on the nuScenes validation set (Caesar et al. 2020), which consists of 150 frame sequences of 20 seconds each. The implementation of our object criticality model exploits the development kit of nuScenes, which is available with open-source license. We extended the development kit, to have it compute the measures from our model alongside the usual measures of the nuScenes object detection challenge. Therefore, any object detector whose output is compatible with nuScenes can be also evaluated using our library. The library is released open source on github, including tutorials that reproduce the experiments described in this paper. We used the nuScene development kit v1.1.2, and we tested for compatibility up to 1.1.7.

5 Experiments and Results

We execute the nine object detectors on the validation set previously described. We compute AP_{crit}, $P_{\mathcal{R}}$ and $R_{\mathcal{S}}$ for different values of $D_{max}, R_{max}, T_{max}$. More specifically, we consider several configurations $(D_{max}, R_{max}, T_{max})$, with $D_{max} \in \{5, 10, \ldots, 50\}$ meters, $R_{max} \in \{5, 10, \ldots, 50\}$ meters, and $T_{max} \in \{2, 4, \ldots 30\}$ seconds. Since distance is measured starting from the center of ego, a distance of 5 meters includes only vehicles very close to ego; 50 meters instead is the maximum distance from ego that is considered in the nuScenes object detection challenge, where objects farther than 50 meters from ego are ignored. Overall, this leads to 1500 configurations $(D_{max}, R_{max}, T_{max})$, repeated for each of the 8 object detectors.

5.1 AP_{crit} and ranking of object detectors

First, we calculate the rankings of detectors based on AP_{crit} for all the 1500 configurations $(D_{max}, R_{max}, T_{max})$. Many of them produced a different ranking with respect to the one based on AP. For example, consider $l = 0.5$ and $l = 4$. When $l = 0.5$, the ranking calculated with AP_{crit} does not match the AP ranking for 567 out of 1500 configurations; for each of these 567 configurations, the difference with respect to the AP ranking is 2 or 4 positions. The whole set of object detectors may change position with respect to the AP ranking, with the exception of the detector in the 7^{th} and 8^{th} position which are always PGD and FCOS, respectively. For $l = 4$, the ranking changes in 1425 out of 1500 configurations, and all the object detectors may change position, including FCOS performing better than PGD. In Table 2, we compare the AP and AP_{crit} ranking of the nine object detectors, for exemplary configurations $(D_{max}, R_{max}, T_{max})$. Noticeably, the object detector with the highest AP, REG1.6, is outperformed by SSNREG and also others when we consider AP_{crit}.

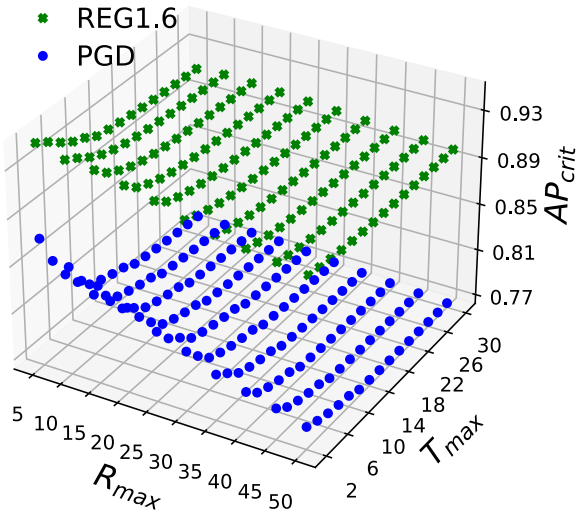

Figure 2: AP_{crit} measured on REG1.6 and PGD with $l = 2$ and $D_{max} = 25$.

To explore trends of AP_{crit}, we select a representative example, that is object detectors REG1.6 ($AP = 0.874$) and PGD ($AP = 0.703$) with $l = 2$. In Figure 2 we show their AP_{crit} values when $D_{max} = 25$, and for different R_{max}, T_{max}. The AP_{crit} of REG1.6 and PGD is higher than the respective APs under the considered configurations. In fact, setting $D_{max} = 25$ reduces the impact of objects farther than 25 meters, which are a significant contribution to misdetections.

We remark that, while studies like Figure 2 are effective to explain the proposed measure, the most suitable configuration $(D_{max}, R_{max}, T_{max})$ should be decided based on the requirements of the target application, and then the object detector with the highest AP_{crit} for such configuration should be selected. In general, higher values of AP_{crit} are achieved with low values of R_{max} and T_{max}; for both REG1.6 and PGD, the maximum AP_{crit} values are obtained with $(D_{max}, R_{max}, T_{max}) = (25, 5, 2)$. Intuitively, low R_{max} and T_{max} reduce the number of vehicles to be considered in our analysis: only those that are really critical for the detection are included. Analogous observations can be derived with the other configurations and object detectors.

5.2 Tradeoff between $P_{\mathcal{R}}$ and $R_{\mathcal{S}}$

To discuss the relations between $P_{\mathcal{R}}$ and $R_{\mathcal{S}}$, we rely on Figure 3 and Figure 4, where we use SSNREG with $l \in \{1, 4\}$. We compute $P_{\mathcal{R}}$ and P for, respectively, $R_{\mathcal{S}}$ and R at steps of 0.01, starting from 0.85. Red crosses represent precision-recall pairs (P, R). Black dots represent $(P_{\mathcal{R}}, R_{\mathcal{S}})$ pairs; these are computed for each configuration $(D_{max}, R_{max}, T_{max})$, thus yielding 1500 black dots for each $R_{\mathcal{S}}$ value. The large blue dots are the $(P_{\mathcal{R}}, R_{\mathcal{S}})$ values achieved using SSNREG with the configuration from Table 2b and Table 2d, while the green triangle are the configuration leading to the highest AP_{crit}, which is $(25, 5, 2)$.

Table 2: AP and AP_{crit} of car detection, for nine object detectors and $l \in \{0.5, 1, 2, 4\}$, ordered by AP. AP_{crit} is computed with $(D_{max}, R_{max}, T_{max})$ amongst the configurations that reported the highest differences between AP_{crit} and AP ranking. Ranking differences are in bold.

(a) $l = 0.5$, $(20, 20, 8)$

Detector	AP	AP_{crit}
FCOS	0.118	0.1995
PGD	0.172	0.2711
SEC	0.677	0.7534
FPN	0.682	**0.7618**
REG400	0.690	**0.7616**
SSN	0.696	0.7717
REGSEC	0.713	0.7773
SSNREG	0.717	**0.7903**
REG1.6	0.722	**0.7895**

(b) $l = 1$, $(20, 25, 10)$

Detector	AP	AP_{CRIT}
FCOS	0.372	0.4857
PGD	0.450	0.5658
SEC	0.796	0.8486
FPN	0.814	**0.8631**
SSN	0.818	**0.8628**
REGSEC	0.825	**0.8728**
REG400	0.827	**0.8703**
SSNREG	0.835	**0.8806**
REG1.6	0.837	**0.8769**

(c) $l = 2$, $(10, 35, 8)$

Detector	AP	AP_{CRIT}
FCOS	0.655	0.7061
PGD	0.703	0.7382
SEC	0.833	0.8611
FPN	0.857	**0.8748**
SSN	0.860	**0.8691**
REGSEC	0.863	**0.8838**
REG400	0.870	**0.8839**
SSNREG	0.872	**0.8912**
REG1.6	0.874	**0.8837**

(d) $l = 4$, $(20, 25, 4)$

Detector	AP	AP_{CRIT}
FCOS	0.804	0.8695
PGD	0.828	0.8860
SEC	0.852	0.9057
FPN	0.872	**0.9202**
REGSEC	0.875	**0.9205**
SSN	0.876	**0.9200**
REG400	0.884	**0.9267**
SSNREG	0.886	**0.9293**
REG1.6	0.889	**0.9264**

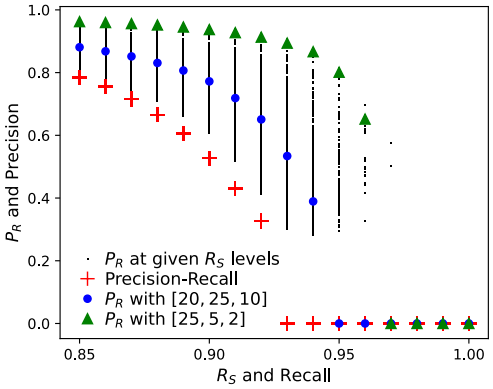

Figure 3: $P_{\mathcal{R}}$, $R_{\mathcal{S}}$, P and R for SSNREG when $R_S \geq 0.85$ and $R \geq 0.85$, with $l = 1$.

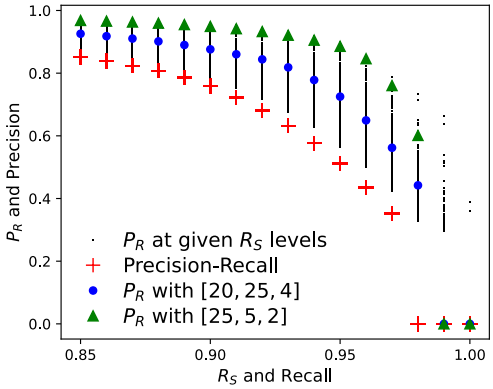

Figure 4: $P_{\mathcal{R}}$, $R_{\mathcal{S}}$, P and R for SSNREG when $R_S \geq 0.85$ and $R \geq 0.85$, with $l = 4$.

We investigate the relations between $P_{\mathcal{R}}$ and $R_{\mathcal{S}}$ for high values of $R_{\mathcal{S}}$ (safety-weighted recall), which are of particular interest in the reference domain of this work. This way we can study the $P_{\mathcal{R}}$ (reliability-weighted precision) that we achieve when safety is enforced thanks to a high $R_{\mathcal{S}}$. This corresponds to answering the question "given a safety target on the detection, what is the possibility of driving the car with good mission reliability, i.e., without being forced to interrupt the driving continuously because of false positives?". Of course, the safest condition would be $R_S = 1$, but $P_{\mathcal{R}}$ is typically 0 in such cases; still, a very high $R_{\mathcal{S}}$ is necessary to enforce safety of the detection.

When the recall R increases, the precision P quickly drops to 0. SSNREG can offer a high recall, i.e., high ability of detecting all the objects, only at the cost of many false positives: this is clearly of little or no use in practice. Instead, if we restrict the scope of the object detector thanks to our model, we reach different conclusions. For example, consider again the case $l = 1$ (Table 2b). Even with $R_{\mathcal{S}} \geq 0.9$, there are some configurations in which $P_{\mathcal{R}} > 0.8$, which is clearly a much more comforting result, showing confidence in the detection at least to some extent.

On the other hand, the best performing triples, represented with the green triangles in Table 2b, may be not practical, because it is computed applying small spatial and temporal distances of the objects from ego. Summarizing, our conclusion on SSNREG can be very different from those we achieve using P and R, when we apply the criteria of $R_{\mathcal{S}}$ and $P_{\mathcal{R}}$.

6 Conclusions and Future Works

We argue that the most used measures for object detection do not match the demands and peculiarities of a safety-critical system. Within the autonomous driving domain, currently adopted measures typically describe how good an object detector is at detecting *all* the objects on the scene, while instead, for the purpose of an autonomous driving system, we are interested in detecting all the objects that will likely interfere with the driving task of the vehicle.

To this end, we propose novel measures that take into account the concepts of safety (detection of dangerous objects, which require immediate reaction, should be prioritized) and reliability (misdetections should not severely disrupt the continuity of the driving task). We build and exercise an object criticality model that performs a rating of the objects, based on the distance from the subject vehicle, the possible colliding trajectory, and the expected time to collision. Amongst

the main results, we show that our judgement on the performance of object detectors may be very different when we consider the detection of i) everything on the scene (as it is usually done), or ii) only the relevant items.

Last, an important implication of our model is that, when safety and reliability issues are considered, the selection of the most suitable object detector strictly depends on its desired use, i.e., on the requirements of the target application. Starting from application requirements, the desired configuration of our model is identified, measures are computed, and the most suitable object detector is selected.

As future work, we are currently working towards training a model to maximise AP_{crit}. Intuitively, the model is intended to reward the detection of objects that are relevant (close and in colliding trajectories), and it is expected instead to be far less effective in the detection of objects that are not relevant for the driving tasks and that do not interfere with the elaboration of the trajectory of ego.

References

Avizienis, A.; Laprie, J.-C.; Randell, B.; and Landwehr, C. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing*, 1(1): 11–33.

Bansal, A.; Singh, J.; Verucchi, M.; Caccamo, M.; and Sha, L. 2021. Risk Ranked Recall: Collision Safety Metric for Object Detection Systems in Autonomous Vehicles. In *2021 10th Mediterranean Conference on Embedded Computing (MECO)*, 1–4. IEEE.

Caesar, H.; Bankiti, V.; Lang, A. H.; Vora, S.; Liong, V. E.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G.; and Beijbom, O. 2020. nuScenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11621–11631.

Contributors. 2020. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d.

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; and Wei, Y. 2017. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, 764–773.

Everingham, M.; Eslami, S. A.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2015. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1): 98–136.

Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338.

Geiger, A.; Lenz, P.; Stiller, C.; and Urtasun, R. 2013. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11): 1231–1237.

Grigorescu, S.; Trasnea, B.; Cocias, T.; and Macesanu, G. 2020. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3): 362–386.

Houston, J.; Zuidhof, G.; Bergamini, L.; Ye, Y.; Chen, L.; Jain, A.; Omari, S.; Iglovikov, V.; and Ondruska, P. 2021. One Thousand and One Hours: Self-driving Motion Prediction Dataset. In Kober, J.; Ramos, F.; and Tomlin, C., eds., *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, 409–418. PMLR.

Lang, A. H.; Vora, S.; Caesar, H.; Zhou, L.; Yang, J.; and Beijbom, O. 2019. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12697–12705.

Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; and Belongie, S. 2017a. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2117–2125.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017b. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; and Pietikäinen, M. 2020. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2): 261–318.

Lyssenko, M.; Gladisch, C.; Heinzemann, C.; Woehrle, M.; and Triebel, R. 2021. From evaluation to verification: Towards task-oriented relevance metrics for pedestrian detection in safety-critical domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 38–45.

Powers, D. M. 2020. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*.

Radosavovic, I.; Kosaraju, R. P.; Girshick, R.; He, K.; and Dollár, P. 2020. Designing Network Design Spaces. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10425–10433.

Salton, G.; and McGill, M. J. 1983. *Introduction to modern information retrieval*. mcgraw-hill.

Sun, P.; Kretzschmar, H.; Dotiwalla, X.; Chouard, A.; Patnaik, V.; Tsui, P.; Guo, J.; Zhou, Y.; Chai, Y.; Caine, B.; Vasudevan, V.; Han, W.; Ngiam, J.; Zhao, H.; Timofeev, A.; Ettinger, S.; Krivokon, M.; Gao, A.; Joshi, A.; Zhang, Y.; Shlens, J.; Chen, Z.; and Anguelov, D. 2020. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2443–2451.

Topan, S.; Leung, K.; Chen, Y.; Tupekar, P.; Schmerling, E.; Nilsson, J.; Cox, M.; and Pavone, M. 2022. Interaction-Dynamics-Aware Perception Zones for Obstacle Detection Safety Evaluation. *arXiv preprint arXiv:2206.12471*.

Trivedi, K. S. 1982. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. Prentice Hall.

Volk, G.; Gamerdinger, J.; Bernuth, A. v.; and Bringmann, O. 2020. A Comprehensive Safety Metric to Evaluate Perception in Autonomous Systems. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 1–8.

Wang, T.; Xinge, Z.; Pang, J.; and Lin, D. 2022. Probabilistic and geometric depth: Detecting objects in perspective. In *Conference on Robot Learning*, 1475–1485. PMLR.

Wang, T.; Zhu, X.; Pang, J.; and Lin, D. 2021. FCOS3D: Fully Convolutional One-Stage Monocular 3D Object Detection. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 913–922. Los Alamitos, CA, USA: IEEE Computer Society.

Yan, Y.; Mao, Y.; and Li, B. 2018. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10): 3337.

Zhu, X.; Ma, Y.; Wang, T.; Xu, Y.; Shi, J.; and Lin, D. 2020. SSN: Shape Signature Networks for Multi-class Object Detection from Point Clouds. In Vedaldi, A.; Bischof, H.; Brox, T.; and Frahm, J.-M., eds., *Computer Vision – ECCV 2020*, 581–597. Cham: Springer International Publishing. ISBN 978-3-030-58595-2.