Bergische Universität Wuppertal

Fakultät für Mathematik und Naturwissenschaften

Institute of Mathematical Modelling, Analysis and Computational Mathematics (IMACM)

Preprint BUW-IMACM 21/22

Tobias Riedlinger, Matthias Rottmann, Marius Schubert und Hanno Gottschalk

Gradient-Based Quantification of Epistemic Uncertainty for Deep Object Detectors

July 12, 2021

http://www.imacm.uni-wuppertal.de

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

Gradient-Based Quantification of Epistemic Uncertainty for Deep Object Detectors

Tobias Riedlinger, Matthias Rottmann, Marius Schubert and Hanno Gottschalk,
School of Mathematics and Natural Sciences, University of Wuppertal,
{riedlinger, rottmann, mschubert, hgottsch}@uni-wuppertal.de

Abstract

Reliable epistemic uncertainty estimation is an essential component for backend applications of deep object detectors in safety-critical environments. Modern network architectures tend to give poorly calibrated confidences with limited predictive power. Here, we introduce novel gradient-based uncertainty metrics and investigate them for different object detection architectures. Experiments on the MS COCO, PASCAL VOC and the KITTI dataset show significant improvements in true positive / false positive discrimination and prediction of intersection over union as compared to network confidence. We also find improvement over Monte-Carlo dropout uncertainty metrics and further significant boosts by aggregating different sources of uncertainty metrics. The resulting uncertainty models generate well-calibrated confidences in all instances. Furthermore, we implement our uncertainty quantification models into object detection pipelines as a means to discern true against false predictions, replacing the ordinary score-threshold-based decision rule. In our experiments, we achieve a significant boost in detection performance in terms of mean average precision. With respect to computational complexity, we find that computing gradient uncertainty metrics results in floating point operation counts similar to those of Monte-Carlo dropout.

Index Terms

deep learning, uncertainty quantification, gradient metrics, object detection

I. Introduction

Deep artificial neural networks (DNNs) designed for tasks such as object detection or semantic segmentation provide a probabilistic prediction on given feature data such as camera images. Modern deep object detection architectures [1, 2, 3, 4] predict bounding boxes for instances of a set of learnt classes on an input image. The so-called objectness or confidence score, indicates the probability of the existence of an object. Throughout this work, we will refer to this quantity which the DNN learns as the "score". We use the term "confidence" more broadly than "score" to refer to quantities which represent the probability of a detection being correct, thus reflecting the model's overall level of competency when presented with a given input

Accurate probabilistic predictions are desirable for applications such as automated surgery or driving. A well-known problem with modern DNNs in general [5, 6, 7], is that their probabilistic predictions are statistically *miscalibrated*, i.e., the score they generate is not reflective of the relative frequency of correct predictions. See fig. 1 for a comparison of different sources of uncertainty in terms of their reliability / conditional accuracy over confidence bins.

For individual detections, miscalibration implies that the given confidence and thus, the entire prediction is unreliable. This, in turn, could conceivably lead to technological breakdown. Over-confident predictions might render an autonomous driving system inoperable by perceiving non-existent instances (false positives / FP). Perhaps even more detrimental, under-confidence may lead to false negative (FN) predictions possibly endangering humans outside autonomous vehicles like pedestrians and cyclists, as well as passengers.

The notion of prediction confidence is intimately linked to measures of uncertainty and the probability estimation of a prediction being accurate. Uncertainty for statistical models, in particular DNNs, can broadly be divided into two types [8] depending on their primary source [9, 10]. Whereas *aleatoric* uncertainty is mainly founded in the stochastic nature of the data generating process, *epistemic* uncertainty stems from the probabilistic nature of sampling data for training, as well, as the choice of model and the training algorithm. The latter is technically reducible by obtaining additional training data.

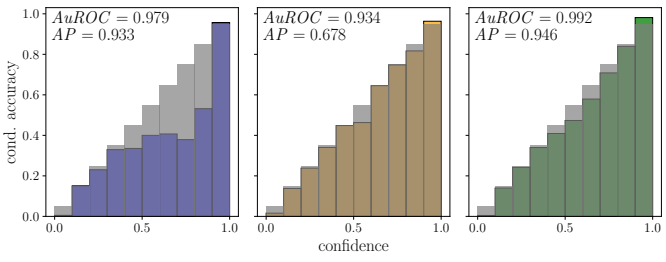

Fig. 1. Reliability plots for score (left), MC dropout uncertainty (center) and our proposed gradient uncertainty metrics G (right). As a reference, we show the optimally calibrated diagonal in gray. See section IV and section D for more details. Model: Faster R-CNN, dataset: KITTI evaluation split.

Nowadays, there are a few established approaches to properly capture uncertainty. The authors of [10] propose to add regression outputs to DNNs in order to generate heteroscedastic models resulting in image-dependent uncertainty measures. The model is trained with a negative log-likelihood loss depending on the uncertainty output. This approach has found adaptations in modern DNNs for a broad range of applications. From a theoretical point of view, Bayesian DNNs [11, 12]

represent an attractive framework for capturing epistemic uncertainty for DNNs by modeling their weights as random variables. Practically, this approach introduces a large computational overhead making its application infeasible for tasks such as object detection. Therefore, in variational inference approaches, weights are sampled from predefined distributions to address this. A standard tool for estimating epistemic uncertainty is Monte-Carlo (MC) dropout variational inference [13, 14], employing several forward passes under dropout. The same idea underlies deep ensemble sampling [15] where separately trained models with the same architecture produce variational forward passes.

In [16] the usage of a learning gradient evaluated at the network's own prediction was proposed to contain epistemic uncertainty information. Since the initial investigation for image classification, the method has attracted attention in the literature. A survey compared gradient metrics to other uncertainty quantification methods in [17] where gradient uncertainty was found to be strongly correlated with the softmax entropy of classification networks. The method has also been applied gainfully to natural language understanding [18] where gradient metrics and deep ensemble uncertainty were aggregated to obtain well-calibrated confidence measures on out-of-distribution data.

The new method we propose here implements gradient-based uncertainty metrics for three modern object detection architectures [2, 3, 4]. We use the obtained metrics for predicting detection quality and compare them in this discipline with the score as well as MC dropout and the output-based framework proposed in [19]. We investigate how these methods can be combined by aggregation for a more comprehensive estimation of prediction uncertainty. Our main contributions are:

- We introduce *novel gradient-based epistemic uncertainty metrics* and show how to implement them for different modern object detection architectures.
- We investigate the performance of gradient metrics in terms of *calibration*, *meta classification* (FP detection) and *meta regression* (prediction of intersection over union IoU with the ground truth) and compare them with other means to quantify epistemic uncertainty. We show that the resulting meta classifiers can be used to improve object detection performance in terms of mean average precision (mAP).
- We provide a *theoretical treatment of the computational complexity* of gradient metrics in comparison with MC dropout.

II. Related work

Recent progress in the estimation of aleatoric uncertainty has been made in [20, 21, 22, 23, 24, 25]. In [20], a two-stage detector is adapted to learn the IoU of its prediction with the ground truth directly. Meanwhile, approaches such as [21, 22, 23, 24, 26, 27] learn the bounding box localization regression for various architectures as Gaussian distributions for each localization variable individually. These approaches aim at learning aleatoric uncertainty from the training dataset in the manner described in [10].

As a tool for estimating epistemic uncertainty, MC dropout sampling has been successfully applied to deep object detection in [28]. The authors of [26] employed MC dropout and Bayesian inference as a replacement of Non-Maximum Suppression (NMS) to get a joint estimation of epistemic and aleatoric uncertainty. Similarly, uncertainty measures of epistemic and aleatoric kind were obtained in [25]. A novel approach to predictive uncertainty estimation in object detection was presented in [19]. The approach is specific to anchor-based object detection architectures, where NMS is used to filter different output boxes indicating the same object instance. The authors of [19] use the set of the filtered boxes for each predicted bounding box to generate uncertainty metrics in a post-processing step. Deep ensemble object detectors were investigated in [29] for a variety of architectures. The authors of [29] accumulate predictions from an ensemble of models and filter from all proposals by NMS treating the entire ensemble as a single detector.

The concept of meta classification for detecting FP and FN on the basis of uncertainty metrics was first explored in [30]. The authors applied the central idea of combining different uncertainty metrics into one and then using the resulting quantity to threshold upon, thereby discriminating true positives (TP) versus FP predictions. Since then, the approach has been applied to semantic segmentation e.g. in [31, 32, 33, 34, 35], instance segmentation in videos [36] and object detection [19, 37]. The common idea in all of these adaptations is the usage of lightweight classification and regression models to explicitly learn the map between uncertainty and the labels TP/FP or between uncertainty and IoU. Additionally, [31] introduces an approach to use the mechanism of detecting FP predictions to improve network accuracy. This is achieved by first increasing the DNN's sensitivity resulting in additional FPs at first and then detecting those via a meta classifier. This method, therefore, uses uncertainty information to improve prediction performance.

Our approach to uncertainty quantification is related to the one explored in [16] which was mentioned in section I. In contrast to previous articles, the present work focuses on generating gradient uncertainty metrics for deep object detectors. In a manner similar to [19], we use meta classification and meta regression to aggregate different gradient-based uncertainty metrics into predictive models and investigate their performance and calibration. We adapt the approach from [31] to the object detection setting and compare the different uncertainty sources in terms of the resulting detection performance.

III. Methods

We fix the notation related to object detection in section A in detail.

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

A. Epistemic uncertainty based on gradients

Generically, given an input $\boldsymbol{x}$, a neural network infers a prediction $\hat{y}(\boldsymbol{x}, \boldsymbol{w})$ given a set of weights $\boldsymbol{w}$. The latter is compared to the ground truth y belonging to $\boldsymbol{x}$ by means of some loss function $\mathcal{L}(\cdot, \cdot)$. Assume that $\hat{y}$ is close to y, i.e., for all practical purposes the prediction can be considered "correct". Then, $\mathcal{L}(\hat{y}, y)$ is small, making only little adjustment in $\boldsymbol{w}$ necessary. This adjustment is proportional to the gradient $g(\boldsymbol{x}, \boldsymbol{w}, y) := \nabla_{\boldsymbol{w}} \mathcal{L}(\hat{y}(\boldsymbol{x}, \boldsymbol{w}); y)$ in standard gradient descent (see (8) in section A). Hence, for correct predictions we expect this quantity to be of comparatively small magnitude. Conversely, when $\hat{y}$ is decidedly wrong, i.e., $\mathcal{L}(\hat{y}, y)$ is large, there is a large adjustment in $\boldsymbol{w}$ required. We then expect $g(\boldsymbol{x}, \boldsymbol{w}, y)$ to have a relatively large magnitude. Note, that this does not necessarily hold for non-convex optimization problems. In the following section, we elaborate more in-depth upon this correspondence.

a) General intuition: In order to use $g(\boldsymbol{x}, \boldsymbol{w}, y)$ as a measure of uncertainty, we assume that an optimally trained network has maximal confidence in its *then correct* prediction $\hat{y}(\boldsymbol{x}, \boldsymbol{w})$. Regarding the loss function, such an optimally trained network sits right at a global minimum of the function $\boldsymbol{w} \mapsto \mathcal{L}(\hat{y}(\boldsymbol{x}, \boldsymbol{w}), y)$. A network which is well-trained in the usual sense has $\boldsymbol{w}$ at least close to a local minimum. Then, $g(\boldsymbol{x}, \boldsymbol{w}, y)$ can still be interpreted as the amount of learning stress or weight adjustment induced on $\boldsymbol{w}$ by training on the data $(\boldsymbol{x}, y)$. However, note that this makes use of the ground truth which is not accessible during inference in all applications.

To assign uncertainty to the prediction $\hat{y}(\boldsymbol{x}, \boldsymbol{w})$, the approach presented in [16] is to replace the ground truth y by the prediction $\hat{y}$. We then disregard its dependency on $\boldsymbol{w}$ when taking derivatives. That is, we use

$$g(\boldsymbol{x}, \boldsymbol{w}, \hat{y}) = \nabla_{\boldsymbol{w}} \mathcal{L}(\hat{y}(\boldsymbol{x}, \boldsymbol{w}), \hat{y}) \tag{1}$$

as the basis for uncertainty metrics where now, derivatives are only taken w.r.t. the first slot of $\mathcal{L}(\cdot, \cdot)$. In addition, we assume that $\hat{y}$ in the second slot takes on the form of a ground truth. For classification tasks, the class distribution $\hat{y}(\boldsymbol{x}, \boldsymbol{w})$ entering $\mathcal{L}$ as the *first argument* is collapsed by an $\arg\max$ to the predicted label before entering $\mathcal{L}$ in the *second argument*. In contrast to $g(\boldsymbol{x}, \boldsymbol{w}, y)$, eq. (1) indicates the network's incentive to adjust $\boldsymbol{w}$ given that its own prediction is already correct.

It is a natural idea to then map $g(\boldsymbol{x}, \boldsymbol{w}, \hat{y})$ to its length or magnitude. To this end, we may employ e.g. some norm $\|\cdot\|$. In our experiments, we investigate the usage of norms on the one hand, but also explore other maps casting vectors to scalars. Additional relevant uncertainty information can be extracted from the gradient $g(\boldsymbol{x}, \boldsymbol{w}, \hat{y})$. In particular, we regard $\min(\cdot)$, $\max(\cdot)$, the arithmetic mean $\mathrm{mean}(\cdot)$ and the standard deviation $\mathrm{std}(\cdot)$ in addition to the 1- and 2-norm $\|\cdot\|_1$ and $\|\cdot\|_2$. Note that $\min(\cdot)$ and $\max(\cdot)$ are closely related to the sup norm $\|\cdot\|_\infty$, depending on the sign of the vector components. Experimentally, we have found that both $\min(\cdot)$ and $\max(\cdot)$ carry relevant predictive uncertainty information.

The gradient (1) as a basis for uncertainty metrics exhibits some additional flexibility for extracting information. We are able to compute uncertainty metrics for each summand of $\mathcal{L}$ individually. Also, we can restrict the amount of variables with respect to which we take the gradient. While all trainable variables $\boldsymbol{w}$ of our model may be used, we can also use only a subset $\boldsymbol{w}_\ell$ of these and utilize the partial gradient $\nabla_{\boldsymbol{w}_\ell} \mathcal{L}(\hat{y}(\boldsymbol{x}, \boldsymbol{w}), \hat{y})$. Here, $\boldsymbol{w}_\ell$ could be all the weights from one particular network layer, one particular filter of a convolutional layer, or even singleton weights. This approach also potentially lets us localize uncertainty within the network architecture, an idea that we mention here but which is not further investigated in this work. We explain in section IV in more detail how we use these facts for our concrete application.

b) Application to object detectors: All $\hat{N}_{\mathrm{out}}$ output bounding boxes potentially contribute to $g(\boldsymbol{x}, \boldsymbol{w}, y)$ in a single optimization step (refer to section B for details). Hence, the gradient as defined in eq. (1) can be viewed as containing uncertainty for the entire prediction $\hat{y}(\boldsymbol{x}, \boldsymbol{w})$ but not for individual boxes $\hat{y}^j(\boldsymbol{x}, \boldsymbol{w})$. Therefore, we regard only one fixed prediction $\hat{y}^j$ and imagine all other predictions erased from $\hat{y}$, so we treat only $\hat{y}^j$ as the *second* argument of $\mathcal{L}$. Suppose, we inserted the entire prediction $\hat{y}(\boldsymbol{x}, \boldsymbol{w})$ as the *first* argument. The result $\mathcal{L}(\hat{y}(\boldsymbol{x}, \boldsymbol{w}), \hat{y}^j)$ measures the mistakes made by the network given that the entire ground truth were $\hat{y}^j$. Potentially all previously learnt boxes will then be punished since they deviate from the "restricted" ground truth $\hat{y}^j$. This happens irrespective of whether they are correct (compared with the real and complete ground truth y) detections on $\boldsymbol{x}$ or not. Then, $g(\boldsymbol{x}, \boldsymbol{w}, \hat{y}^j)$ adjusts $\boldsymbol{w}$ such that correctly predicted boxes are un-learnt. This phenomenon is obviously undesired in light of our design motivation for gradient uncertainty. Our proposal to resolve this issue is to adjust the first argument of $\mathcal{L}$ by only including boxes indicating the same object as $\hat{y}^j$ ("candidate boxes", see section A):

$$g^{\mathrm{cand}}(\boldsymbol{x}, \boldsymbol{w}, \hat{y}^j) := \nabla_{\boldsymbol{w}} \mathcal{L}\left(\mathrm{cand}[\hat{y}^j](\boldsymbol{x}, \boldsymbol{w}), \hat{y}^j\right). \tag{2}$$

It follows that the loss function computed in this way only punishes instances belonging to a cluster of instances located close to $\hat{y}^j$ with the same assigned class.

c) Computational complexity of gradient metrics: Considering fully convolutional DNNs, we address the computational cost of gradient uncertainty metrics. Similarly to [38, section 20.6], we regard a neural network with $T \in \mathbb{N}$ layers, where the activation in layer $t \in \{1, \ldots, T\}$ is a stack of feature maps $\phi_t \in \mathbb{R}^{h \times w \times k_t}$. Here, $h, w, k_t \in \mathbb{N}$ are feature map height, width and number of layer channels, respectively. The activation ϕ_t is obtained by convolution of ϕ_{t-1} with a stack $K_t \in \mathbb{R}^{k_{t-1} \times k_t \times (2s_{t-1})^2}$ of filter matrices where $s_{t-1} \in \mathbb{N}$ is the spatial expanse of the filter matrices (oftentimes 0 or 1). We consider the loss function to be computed from the feature maps ϕ_T of the last layer with ground truth feature map γ which is considered fixed. The loss function can then be regarded as a function $\ell_T : \phi_T \mapsto \mathcal{L}(\phi_T, \gamma)$. This way, the loss function implicitly also depends on all

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

activations ϕ_t for $t < T$. We consider an explicit computation of the gradient by backpropagation and count the necessary floating point operations (FLOP) to compute $\nabla_{K_t}\mathcal{L}$. We assume that binary decisions in the computation can be neglected and for the sake of comparison with MC dropout uncertainty, that one full forward pass through the network has already been performed. For this section we use the symbol D to denote total derivatives and D_1 for the differential w.r.t. to the first variable. The last layer gradient $\nabla_{K_T}\mathcal{L} = D_1\mathcal{L}|_{\phi_T} \cdot \nabla_{K_T}\phi_T$ is already dependent on the derivative $D_1\mathcal{L}|_{\phi_T}$, the concrete form of which depends on the exact form of $\mathcal{L}$. We defer the treatment of this term to section C. Here, we present a statement of larger generality on the computational cost of gradients, when $D_1\mathcal{L}|_{\phi_T}$ has been evaluated once. We know the feature map representation γ^j for any predicted box $\hat{y}^j$. We denote the mask μ^j over ϕ_T such that $\mu^j\phi_T$ are the feature map representations of the candidate boxes $\mathrm{cand}[\hat{y}^j]$ of $\hat{y}^j$.

Theorem 1. *In the setting explained above, the number of FLOP required to compute the gradient $\nabla_{K_T}\mathcal{L}(\mu^j\phi_T(K_T), \gamma^j)$ is $\mathcal{O}(k_T hw + k_T k_{T-1}(2s_T+1)^4)$. Similarly, for earlier layers, i.e., $\nabla_{K_t}\mathcal{L}(\mu^j\phi_T(K_t), \gamma^j)$, we have $\mathcal{O}(k_{t+1}k_t + k_t k_{t-1})$, provided that we have previously computed the gradient for the consecutive layer $t+1$. Performing MC dropout with dropout on ϕ_{T-1} requires $\mathcal{O}(k_T k_{T-1} hw)$ FLOP per dropout sample.*

In section C, we provide a proof of these results and further detail such as the proportionality constants. Note, that dropout anywhere earlier in the forward pass leads to a considerable increase in FLOP. The complexity of computing $\nabla_{K_T}\mathcal{L}$ and $\nabla_{K_{T-1}}\mathcal{L}$ for a number of boxes is, therefore, about on par with the complexity of MC dropout sampling in light of this fact. Note, however that gradient metrics need to be iteratively computed whereas MC dropout runs in parallel.

B. Meta classification and meta regression

Meta classification denotes the task of discriminating the truth value of a prediction on the basis of uncertainty metrics. In the context of object detection, we classify individual boxes as TP or FP. While the truth value TP/FP is binary, in object detection we also determine prediction quality in terms of the maximum IoU with the ground truth. Capturing the relationship between uncertainty metrics and IoU is the aim of *meta regression*. The two tasks are closely related to the motivation of curing the short-comings of the score and are explored in section IV. Experiments from [19] indicate that the relation between uncertainty metrics and the labels TP/FP or IoU is highly non-linear. We restrict our attention to gradient boosting [39] models which have proven to effectively capture this non-linearity. Uncertainty metrics serve as co-variables for such a model, and we compare their meta classification performance in terms of area under receiver operating characteristic ($AuROC$) and average precision (AP) [40] as well as their meta regression performance in terms of R^2. We train meta classifiers and meta regression models on uncertainty metrics of boxes *post NMS*. This way, we obtain confidence estimates for boxes which would only be discarded based on their score.

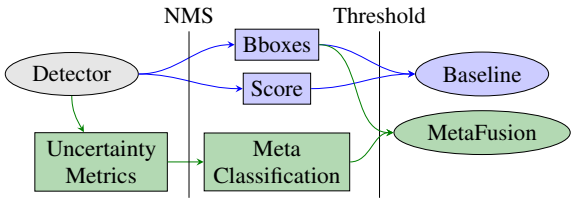

Fig. 2. Schematic sketch of the baseline detection pipeline and the alternative MetaFusion pipeline for an object detector.

a) MetaFusion: As a further application of uncertainty metrics, we investigate an approach inspired by [31]. The authors of [31] use computed uncertainty metrics and a meta classifier to improve prediction performance by reducing the amount of FP predictions. The adaptation to object detection is straight-forward in that we compare the standard detection pipeline with a detection based on meta classification. Whereas state-of-the-art object detectors use score thresholding (fig. 2) in addition to NMS, we propose to base a threshold on meta classification probabilities. Well-performing meta classifiers (which accurately detect FPs) and an increase in detection sensitivity offer a way to "trade" uncertainty information for detection performance. Note, that in most object detection pipelines, score thresholding is carried out before NMS. We choose to interchange them here as they commute for the baseline approach. The resulting predictions are compared in terms of mean Average Precision (mAP [41]).

IV. Experiments

We describe our numerical methods and experimental findings for the object detection datasets Pascal VOC [41], MS COCO [42] and KITTI [43]. The splits for training and evaluation we used are indicated in table I. Since there is no designated test split in KITTI which provides a ground truth, we take 2000 randomly chosen images from the annotated training split for testing and only used the complementary split for training (~ 5500 images).

a) Implementation details: For our experiments, we employ three common object detection architectures, namely YOLOv3 with Darknet53 backbone [2], Faster R-CNN [3] and RetinaNet [4], each with a ResNet50FPN [44] backbone. We started from PyTorch [45] reimplementations,[1], added dropout and trained from scratch on the datasets in table I.

[1]YOLOv3 from `https://github.com/westerndigitalcorporation/YOLOv3-in-PyTorch` Faster R-CNN and RetinaNet from the the `torchvision` library

TABLE I

Left: NUMBER OF LAYERS AND LOSSES UTILIZED AND RESULTING NUMBERS OF GRADIENT METRICS PER BOX. MULTIPLICATION IN # LAYERS DENOTES PARALLEL OUTPUT STRANDS OF THE RESP. DNN NOT GENERATING ADDITIONAL GRADIENTS. *Right*: SPLITS FOR TRAINING AND EXPERIMENTAL EVALUATION FOR THE INVESTIGATED DATASETS.

Architecture	# layers	# Losses	# gradients
YOLOv3	$2(\times 3)$	3	6
Faster R-CNN	$2(\times 4)$	4	8
RetinaNet	$2(\times 2)$	2	4

Dataset	training	evaluation	# eval images
VOC	2012 train	2007 train	4952
COCO	2017 train	2017 val	5000
KITTI	random part of training	complement part of training	2000

We compute gradients layer-wise by standard backpropagation. In [16], it was reported that the last layers give rise to the most informative gradients which is congruent with our experience. For YOLOv3, we use the last two convolutional (conv) layers of each of the three detection arms. In Faster R-CNN, for the region proposal network we use the last two convolutional layers and for the region of interest (RoI) head we use the two fully connected (fc) classification and regression layers. For RetinaNet, we employ the last two conv layers of the classification and regression head, respectively. We install dropout layers in-between those conv and fc layers in each model. MC dropout inference and gradient computation is always done with the same weights, trained under dropout. Per inference, we always compute 30 dropout samples.

We exploit the split of $\mathcal{L}$ into different terms for localization, score and classification (eq. (7) in section A), by computing partial gradients for each contribution to the loss function in order to increase the amount of information obtained. This results in the number of gradients computed indicated in table I. From each gradient, we extract information via the following functions:

$$\{\min(\cdot), \max(\cdot), \mathrm{mean}(\cdot), \mathrm{std}(\cdot), \|\cdot\|_1, \|\cdot\|_2\} \tag{3}$$

In order to save computational effort, we compute gradient metrics not for all predicted bounding boxes. We use a small score threshold of 10^{-4} (KITTI, Pascal VOC), resp. 10^{-2} (COCO) as a pre-filter. On average, this produces ~ 100 predictions per image. For gradient boosting models, we employ the XGBoost library [46] with 30 estimators (otherwise standard settings).

b) Meta classification: We train gradient boosting classifiers on different sets of uncertainty metrics. For comparison of meta classification performance, we compute the mean $AuROC$ and average precision over 10-fold image-wise cross validation (cv). The entries $\|\cdot\|_2$ and $\|\cdot\|_1$+$\|\cdot\|_2$ denote the respective maps used to extract gradient information. The largest set of gradient metrics we are considering, i.e., all of (3) is indicated by "G". For comparison and reference, we also state the respective performance of the MetaDetect ("MD", [19]) metrics.

Table II shows the results for YOLOv3. Note, that AP refers not to the object detector but the meta classifier. We see that $\|\cdot\|_2$ yields better results than the score in all cases except for AP on the KITTI dataset, where they differ by 0.01 AP points. As an enveloping model of $\|\cdot\|_2$, $\|\cdot\|_1$+$\|\cdot\|_2$ is better than the score across the board. The best purely gradient-based models G achieve a consistently better trend of AP compared to MC. The combinations G+MC, G+MD and finally, G+MD+MC show further relative boosts as compared to their individual components, indicating *mutual non-redundancy* between G, MC and MD. We find congruent results for Faster R-CNN and RetinaNet, see section E.

c) Calibration: We evaluate the calibration of the obtained confidences by computing the standard calibration metrics shown in table III (see section D for details) by employing 10 bins of width 0.1 and 10-fold cv as before except for the score which is deterministic. We omit the ECE metric here, as we regard it as less informative but we list it in section D. Table III indicates that all meta classifiers are always better calibrated than the score in both of the two metrics MCE and ACE. MC dropout, $\|\cdot\|_2$ and $\|\cdot\|_1$+$\|\cdot\|_2$ show especially small miscalibration across datasets. The reliability diagrams in fig. 1 of section I show the respective accuracy per bin for three of the entries for Faster R-CNN on KITTI. Meta classifiers show very little mis-calibration as compared to the overconfident score. A full table of calibration errors for all our detectors can be found in section E.

d) Pedetrian detection: Restricting ourselves to the KITTI class "Pedestrian" only, we compare score thresholding and meta classification in fig. 3. By thresholding on score or meta classification probabilities (MC, G and G+MD+MC) from the indicated metrics, we count FP and FN for a number of decision thresholds. The latter are sweeped between 0 (bottom right in fig. 3) and 1 (top left) in increments of 10^{-4} to account for fine steps. This is especially important for the score where a significant part of all predicted Pedestrian instances have scores between 10^{-4} and $5 \cdot 10^{-4}$. We have also indicated two lines of constant total (FP + FN) error in dashed gray showing that the score achieves the best total error on the Pedestrian class at around 350 errors and the models G and G+MD+MC not far behind. However, in safety-critical applications of uncertainty estimation, not all errors are equally important. Employing an object detector in an autonomous vehicle, might demand for a reduction in FN predictions while still keeping the corresponding FPs reasonably low. Regarding the lower right part of fig. 3, we see that at around 1100 FPs, the meta classification models MC, G and G+MD+MC reduce the number of FNs to almost zero. The decision made by the score, in contrast, has more than 50 FNs at the same FP "cost". Alternatively, we might be

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

TABLE II
META CLASSIFICATION PERFORMANCE IN TERMS OF $AuROC$ AND AP PER CONFIDENCE MODEL OVER 10-FOLD CV (std IN PARENTHESES). MODEL: YOLOV3.

	Pascal VOC		COCO		KITTI	
	$AuROC$	AP	$AuROC$	AP	$AuROC$	AP
Score	0.8977 (0.0005)	0.6934 (0.0008)	0.8382 (0.0003)	0.6418 (0.0004)	0.9653 (0.0005)	0.9687 (0.0003)
MC	0.9530 (0.0002)	0.7614 (0.0018)	0.8956 (0.0002)	0.6653 (0.0010)	0.9760 (0.0007)	0.9717 (0.0010)
$\|\cdot\|_2$	0.9368 (0.0004)	0.7563 (0.0018)	0.8605 (0.0004)	0.6425 (0.0006)	0.9731 (0.0005)	0.9686 (0.0010)
$\|\cdot\|_1$+$\|\cdot\|_2$	0.9399 (0.0004)	0.7628 (0.0020)	0.8736 (0.0003)	0.6602 (0.0005)	0.9765 (0.0004)	0.9722 (0.0008)
G	0.9457 (0.0006)	0.7831 (0.0016)	0.8807 (0.0003)	0.6962 (0.0007)	0.9804 (0.0003)	0.9781 (0.0006)
MD	0.9443 (0.0003)	0.7812 (0.0009)	0.8716 (0.0004)	0.6941 (0.0007)	0.9823 (0.0002)	0.9806 (0.0002)
MD+MC	0.9609 (0.0002)	0.8217 (0.0010)	0.9088 (0.0003)	0.7426 (0.0004)	0.9837 (0.0002)	0.9819 (0.0004)
G+MC	0.9645 (0.0003)	0.8319 (0.0013)	0.9146 (0.0003)	0.7471 (0.0005)	0.9833 (0.0004)	0.9813 (0.0004)
G+MD	0.9521 (0.0003)	0.8022 (0.0013)	0.8862 (0.0004)	0.7150 (0.0008)	0.9850 (0.0003)	0.9837 (0.0004)
G+MD+MC	**0.9652 (0.0003)**	**0.8373 (0.0016)**	**0.9157 (0.0004)**	**0.7566 (0.0008)**	**0.9859 (0.0002)**	**0.9845 (0.0002)**

TABLE III
MAXIMUM (MCE) AND AVERAGE (ACE) CALIBRATION ERROR PER CONFIDENCE MODEL OVER 10-FOLD CV (std IN PARENTHESES EXCEPT FOR RAW SCORE WHERE NO META CLASSIFIER IS TRAINED). MODEL: YOLOV3.

	Pascal VOC		COCO		KITTI	
	MCE	ACE	MCE	ACE	MCE	ACE
Score	0.237	0.108	0.065	0.041	0.348	0.227
MC	0.035 (0.006)	0.013 (0.001)	0.024 (0.004)	0.011 (0.001)	**0.040 (0.017)**	**0.014 (0.004)**
$\|\cdot\|_2$	0.033 (0.006)	0.014 (0.002)	**0.013 (0.004)**	**0.004 (0.001)**	0.046 (0.011)	0.019 (0.003)
$\|\cdot\|_1$, $\|\cdot\|_2$	**0.031 (0.005)**	**0.012 (0.001)**	**0.013 (0.003)**	0.006 (0.001)	0.058 (0.023)	0.022 (0.006)
G	0.051 (0.011)	0.021 (0.003)	0.036 (0.006)	0.014 (0.001)	0.086 (0.017)	0.038 (0.004)
MD	0.048 (0.014)	0.018 (0.002)	0.033 (0.005)	0.014 (0.001)	0.078 (0.014)	0.028 (0.004)
G+MD+MC	0.062 (0.012)	0.027 (0.003)	0.041 (0.004)	0.018 (0.001)	0.088 (0.022)	0.044 (0.004)

interested in allowing e.g. at most 50 FNs across the entire dataset.[2] Both, MC and G can reduce the score's 1200 FPs to around 600–700 FPs. G+MD+MC brings the number of FPs down to around 450, far less than half the FPs of the score.

e) Meta regression: We train a gradient boosting regression on the same sets of metrics as before and evaluate their cross-validated performance. The quality of the obtained predictions is evaluated in terms of the coefficient of determination R^2 which we have listed in table IV. We find the same qualitative behavior as for meta classification in table II. Again, our model G performs better than MC dropout and score across the board and is roughly on par with MetaDetect. Combining different sources of uncertainty metrics leads to additional, significant boosts with the best model gaining between 30% and 40% of the winnable R^2 range as compared to score in all cases. Meta regression on Faster R-CNN and RetinaNet show similar relative improvements over the score (see table VII in section E). Figure 4 shows prediction samples for four of the regressions where the left-most

[2]Our evaluation split contains a total of 1152 Pedestrian instances.

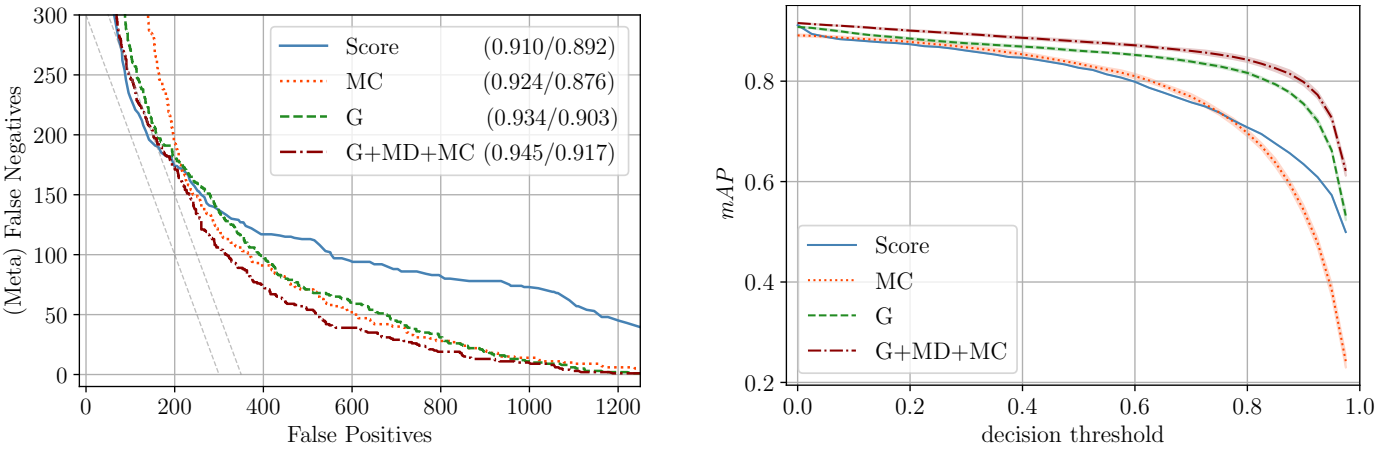

Fig. 3. *Left*: meta classification for the category "Pedestrian". Curves obtained sweeping decision threshold on score or meta classification probability for MC dropout, gradient metrics (G) or the combined model (G+MD+MC). Numbers in parentheses denote ($AuROC/AP$) of the resp. classification. *Right*: score baseline and MetaFusion mAP. Model: YOLOv3, dataset: KITTI.

Preprint – Preprint – Preprint – Preprint – Preprint

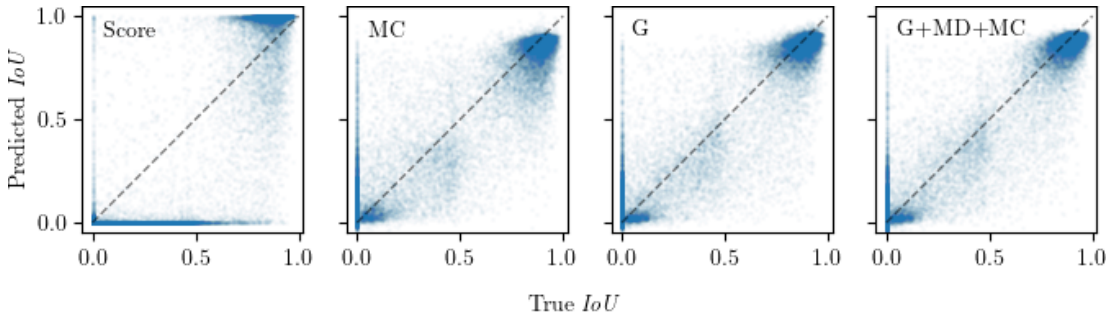

Fig. 4. Scatter plots for samples of Score and meta regression based on MC dropout, gradient metrics G and the combination model G+MD+MC. We draw the optimal diagonal for reference. Model: YOLOv3, dataset: KITTI evaluation split.

panel shows the raw score as opposed to the respective entry in table IV which shows meta regression based on the score. TPs are located in the top right of each panel (high confidence and high IoU with the ground truth), FNs at the bottom right (low confidence but high IoU) and FPs at the top left (high confidence and low / zero IoU). In the left-most plot depicting results for score, we find large amounts of FNs and strong deviations from the diagonal in the TP region. The cluster in the top right above the diagonal shows a very high concentration of "overconfident" TP boxes which is absent in the meta regression models. The latter show from left to right increasing concentration around the diagonal.

TABLE IV
META REGRESSION PERFORMANCE IN TERMS OF R^2 PER CONFIDENCE MODEL OVER 10-FOLD CV (std IN PARENTHESES). MODEL: YOLOV3.

	Pascal VOC	COCO	KITTI
Score	0.4539 (0.0005)	0.3473 (0.0004)	0.7886 (0.0005)
MC	0.5966 (0.0010)	0.4586 (0.0007)	0.8210 (0.0011)
$\|\cdot\|_2$	0.5849 (0.0012)	0.4146 (0.0005)	0.8121 (0.0005)
$\|\cdot\|_1+\|\cdot\|_2$	0.5951 (0.0010)	0.4422 (0.0010)	0.8341 (0.0013)
G	0.6164 (0.0012)	0.4804 (0.0008)	0.8540 (0.0011)
MD	0.5899 (0.0014)	0.4640 (0.0011)	0.8588 (0.0010)
MD+MC	0.6552 (0.0009)	0.5422 (0.0010)	0.8661 (0.0007)
G+MC	0.6799 (0.0013)	0.5543 (0.0007)	0.8652 (0.0012)
G+MD	0.6345 (0.0021)	0.5052 (0.0014)	0.8770 (0.0011)
G+MD+MC	**0.6858 (0.0009)**	**0.5682 (0.0007)**	**0.8805 (0.0010)**

f) MetaFusion: The approach described in section III-B can easily be implemented in the way illustrated in fig. 2. Both, score thresholding and meta classification constitute decision rules on whether to regard a box that has survived the NMS filter as a true or a false prediction, similar to the distinction made on the left in fig. 3. For the resulting prediction, we compute the mAP metric as a measure for the prediction quality across the respective evaluation dataset. As the actual prediction of an object detector depends on either the score threshold or in the MetaFusion case on the meta classification probability threshold, we obtain curves by sweeping the decision threshold between 0 and 1 in steps of 0.025. On the right in fig. 3, we compare the default pipeline Score against MetaFusion pipelines based on MC, G and G+MD+MC. The results have been obtained by 10-fold cv with standard error bands drawn in shaded regions. Note, that the evaluation of the score baseline is deterministic, so it does not have any variance associated with it. We see that for comparatively high decision thresholds ≥ 0.5, the MetaFusion models G and G+MD+MC achieve much higher mAP than the score baseline, with the model based on the larger set of metrics always about 2 percentage points ahead. For decision thresholds smaller than 0.4, the curves naturally approach roughly the same mAP since the thresholding does not filter many boxes anymore and the mAP only differs in the ranking of the boxes, i.e., the order used to construct the class-wise precision-recall curves. However, until the last thresholding step at 0.025, G and "G, MD, MC" still lead to better detection performance with a boost of about 1-2 percentage points in mAP. Interestingly, MC dropout, only performs around as good as the score baseline for thresholds smaller than 0.75. Across all decision thresholds, the gradient metrics-based MetaFusion pipeline achieves better mAP than the MC-based one. We find analogous results also on other datasets (section E).

V. CONCLUSION AND OUTLOOK

Applications of modern DNNs in safety-critical environments demand high performance on the one hand, but also reliable confidence estimation indicating where a model is not competent. Since network-intrinsic confidence is oftentimes mis-calibrated for well-performing DNNs, there is a need for alternative mechanisms and sources of uncertainty information.

We introduced the natural extension (2) of the gradient-based uncertainty metrics introduced by [16] from the image classification setting to object detection, thereby investigating a novel approach to epistemic uncertainty quantification. Equation (2) can in principle be augmented to fit any DNN inferring and learning on an instance-based logic. Gradient-based uncertainty metrics yield well-calibrated, powerful predictors and can be combined with other sources of epistemic uncertainty for additional performance. The predictive capability of gradient uncertainty-based meta classifiers carries over to yield improved object detection pipelines when implemented as a decision mechanism.

Adaptations to 3D-bounding box detection and instance segmentation are planned for our method. As additional applications, meta classifiers are being investigated as a means to estimate dataset labeling quality. Meta regression models predict intersection over union with the ground truth to considerable accuracy, indicating potential as a query mechanism for active learning cycles.

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

An implementation of our method is publicly available at *https://github.com/tobiasriedlinger/gradient-metrics-od*. Find a visual demonstration of meta regression based on gradient uncertainty metrics at *https://youtu.be/L4oVNQAGiBc*.

a) Foreseeable impact: While we see great benefits in increasing the safety of human lives through improved confidence estimation, we also note that the usage of machine learning models in confidence calibration based on gradient information adds another failure mode, which, if applied in automated driving, could lead to fatal consequences, if false predictions of confidence occur. It is, therefore, necessary to be aware of the low technology readiness level of the method introduced here. Despite the evidence provided that our method can significantly reduce the number of false negative pedestrians of an object detector for certain datasets, application of our method in a safety-critical context requires extensive testing to demonstrate robustness, e.g. with respect to domain shifts, prior to any integration into applications.

ACKNOWLEDGMENT

The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project "Methoden und Maßnahmen zur Absicherung von KI basierten Wahrnehmungsfunktionen für das automatisierte Fahren (KI-Absicherung)". The authors would like to thank the consortium for the successful cooperation. Furthermore, we gratefully acknowledge financial support by the state Ministry of Economy, Innovation and Energy of Northrhine Westphalia (MWIDE) and the European Fund for Regional Development via the FIS.NRW project BIT-KI, grant no. EFRE-0400216. The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC).

REFERENCES

[1] Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[2] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[3] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[4] Tsung-Yi Lin et al. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[5] Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).

[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[7] Chuan Guo et al. "On calibration of modern neural networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1321–1330.

[8] Eyke Hüllermeier and Willem Waegeman. "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods". In: *Machine Learning* (2021), pp. 1–50.

[9] Yarin Gal. "Uncertainty in deep learning". PhD thesis. University of Cambridge, 2016.

[10] Alex Kendall and Yarin Gal. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/2650d6089a6d640c5e85b2b88265dc2b-Paper.pdf.

[11] John S Denker and Yann LeCun. "Transforming neural-net output levels to probability distributions". In: *Proceedings of the 3rd International Conference on Neural Information Processing Systems*. 1990, pp. 853–859.

[12] David JC MacKay. "A practical Bayesian framework for backpropagation networks". In: *Neural computation* 4.3 (1992), pp. 448–472.

[13] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[14] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059.

[15] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, 6405–6416. ISBN: 9781510860964.

[16] Philipp Oberdiek, Matthias Rottmann, and Hanno Gottschalk. "Classification uncertainty of deep neural networks based on gradient information". In: *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*. Springer. 2018, pp. 113–125.

[17] Niclas Ståhl et al. "Evaluation of Uncertainty Quantification in Deep Learning". In: *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer. 2020, pp. 556–568.

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

[18] Vishal Thanvantri Vasudevan, Abhinav Sethy, and Alireza Roshan Ghias. "Towards better confidence estimation for neural models". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 7335–7339.

[19] Marius Schubert, Karsten Kahl, and Matthias Rottmann. "MetaDetect: Uncertainty Quantification and Prediction Quality Estimates for Object Detection". In: *arXiv preprint arXiv:2010.01695* (2020).

[20] Borui Jiang et al. "Acquisition of localization confidence for accurate object detection". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 784–799.

[21] Jiwoong Choi et al. "Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 502–511.

[22] Youngwan Lee et al. "Localization Uncertainty Estimation for Anchor-Free Object Detection". In: *arXiv preprint arXiv:2006.15607* (2020).

[23] Michael Truong Le et al. "Uncertainty estimation for deep neural object detectors in safety-critical applications". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 3873–3878.

[24] Yihui He et al. "Bounding box regression with uncertainty for accurate object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2888–2897.

[25] Florian Kraus and Klaus Dietmayer. "Uncertainty estimation in one-stage object detection". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 53–60.

[26] Ali Harakeh, Michael Smart, and Steven L Waslander. "Bayesod: A bayesian approach for uncertainty estimation in deep object detectors". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 87–93.

[27] Jiwoong Choi et al. "Uncertainty-based Object Detector for Autonomous Driving Embedded Platforms". In: *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE. 2020, pp. 16–20.

[28] Dimity Miller et al. "Dropout sampling for robust object detection in open-set conditions". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 3243–3249.

[29] Zongyao Lyu et al. "Probabilistic Object Detection via Deep Ensembles". In: *European Conference on Computer Vision*. Springer. 2020, pp. 67–75.

[30] Dan Hendrycks and Kevin Gimpel. "A baseline for detecting misclassified and out-of-distribution examples in neural networks". In: *arXiv preprint arXiv:1610.02136* (2016).

[31] Robin Chan et al. "MetaFusion: Controlled False-Negative Reduction of Minority Classes in Semantic Segmentation". In: *arXiv preprint arXiv:1912.07420* (2019).

[32] Kira Maag, Matthias Rottmann, and Hanno Gottschalk. "Time-dynamic estimates of the reliability of deep semantic segmentation networks". In: *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE. 2020, pp. 502–509.

[33] Matthias Rottmann et al. "Detection of False Positive and False Negative Samples in Semantic Segmentation". In: *Proceedings of the 23rd Conference on Design, Automation and Test in Europe*. DATE '20. Grenoble, France: EDA Consortium, 2020, 1351–1356. ISBN: 9783981926347.

[34] Matthias Rottmann et al. "Prediction error meta classification in semantic segmentation: Detection via aggregated dispersion measures of softmax probabilities". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–9.

[35] Matthias Rottmann and Marius Schubert. "Uncertainty measures and prediction quality rating for the semantic segmentation of nested multi resolution street scene images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019.

[36] Kira Maag et al. "Improving Video Instance Segmentation by Light-weight Temporal Uncertainty Estimates". In: *arXiv preprint arXiv:2012.07504* (2020).

[37] Kamil Kowol. et al. "YOdar: Uncertainty-based Sensor Fusion for Vehicle Detection with Camera and Radar Sensors". In: *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,* INSTICC. SciTePress, 2021, pp. 177–186. ISBN: 978-989-758-484-8. DOI: 10.5220/0010239301770186.

[38] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[39] Yoav Freund and Robert E Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Journal of computer and system sciences* 55.1 (1997), pp. 119–139.

[40] Jesse Davis and Mark Goadrich. "The relationship between Precision-Recall and ROC curves". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 233–240.

[41] M. Everingham et al. "The Pascal Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338.

[42] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[43] Andreas Geiger et al. "The KITTI vision benchmark suite". In: *URL http://www. cvlibs. net/datasets/kitti* 2 (2015).

[44] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
[45] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
[46] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.
[47] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.7 (2011).
[48] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).
[49] Lukas Neumann, Andrew Zisserman, and Andrea Vedaldi. "Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection". In: *NIPS 2018 Workshop MLITS*. 2018.
[50] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. "Obtaining well calibrated probabilities using bayesian binning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 29. 1. 2015.
[51] Fabian Kuppers et al. "Multivariate Confidence Calibration for Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 326–327.

APPENDIX A
OBJECT DETECTION

We regard the task of 2D bounding box detection on camera images. Here, the detection

$$\hat{y}(\boldsymbol{x}, \boldsymbol{w}) = (\hat{y}^1(\boldsymbol{x}, \boldsymbol{w}), \dots, \hat{y}^{\hat{N}_{\boldsymbol{x}}}(\boldsymbol{x}, \boldsymbol{w})) \in \mathbb{R}^{\hat{N}_{\boldsymbol{x}} \times (4+1+C)} \tag{4}$$

on an input image $\boldsymbol{x}$, depending on model weights $\boldsymbol{w}$, consists of a number $\hat{N}_{\boldsymbol{x}} \in \mathbb{N}$ (dependent on the input $\boldsymbol{x}$) of instances $\hat{y}^j$. We give a short account of its constituents. Each instance

$$\hat{y}^j = (\hat{\xi}^j, \hat{s}^j, \hat{p}^j) \in \mathbb{R}^{4+1+C} \tag{5}$$

consists of localizations $\hat{\xi}^j = (\hat{\mathrm{x}}^j, \hat{\mathrm{y}}^j, \hat{\mathrm{w}}^j, \hat{\mathrm{h}}^j)$ encoded e.g. as center coordinates $\hat{\mathrm{x}}$, $\hat{\mathrm{y}}$ together with width $\hat{\mathrm{w}}$ and height $\hat{\mathrm{h}}$. Moreover, a list of $\hat{N}_{\boldsymbol{x}}$ integers $\hat{\kappa} \in \{1, \dots, C\}$ represents the predicted categories for the object found in the respective boxes out of a pre-determined fixed list of $C \in \mathbb{N}$ possible categories. Usually, $\hat{\kappa}$ is obtained as the $\arg\max$ of a learnt probability distribution $\hat{p} = (\hat{p}_1, \dots, \hat{p}_C) \in (0,1)^C$ over all C categories. Finally, $\hat{N}_{\boldsymbol{x}}$ scores $\hat{s} \in (0,1)$ indicate the probability of each box being correct. The predicted $\hat{N}_{\boldsymbol{x}}$ boxes are obtained by different filtering mechanisms from a fixed number N_{out} (usually about 10^5 to 10^6) of output boxes. The latter are the regression and classification result of pre-determined "prior" or "anchor boxes". Predicted box localization $\hat{\xi}$ is usually learnt as offsets and width- and height scaling of fixed anchor boxes [2, 4] or region proposals [3] (see section B). The most prominent examples of filtering mechanisms are *score thresholding* and *Non-Maximum Suppression*. By score thresholding we mean only allowing boxes which have $\hat{s} \geq \varepsilon_s$ for some fixed threshold $\varepsilon_s \geq 0$.

a) Non-Maximum Suppression (NMS): NMS is an algorithm allowing for different output boxes that have the same class and significant mutual overlap (meaning they are likely to indicate the same visible instance in $\boldsymbol{x}$) to be reduced to only one box. Overlap is usually quantified as *intersection over union* (IoU). For two bounding boxes A and B, their intersection over union is

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{6}$$

i.e., the ratio of the area of intersection of two boxes and the joint area of those two boxes, where 0 means no overlap and 1 means the boxes have identical location and size. Maximal mutual IoU between a predicted box $\hat{y}^j$ and ground truth boxes y is also used to quantify the quality of the prediction of a given instance.

We call an output instance $\hat{y}^i$ "candidate box" for another box $\hat{y}^j$ if it fulfills the following requirements:

1) score ($\hat{s}^i \geq \varepsilon_s$ above a chosen, fixed threshold ε_s)
2) identical class $\hat{\kappa}^i = \hat{\kappa}^j$
3) large mutual overlap $IoU(\hat{y}^i, \hat{y}^j) \geq \varepsilon_{IoU}$ for some fixed threshold $\varepsilon_{IoU} \geq 0$ (a widely accepted choice which we adopt is $\varepsilon_{IoU} = 0.5$).

We denote the set of output candidate boxes for $\hat{y}^j$ by $\mathrm{cand}[\hat{y}^j]$. In NMS, all output boxes are sorted by their score in descending order. Then, the box with the best score is selected as a prediction and all candidates for that box are deleted from the ranked list. This is done until there are no boxes with $\hat{s} \geq \varepsilon_s$ left. Thereby selected boxes form the $N_{\boldsymbol{x}}$ predictions.

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

b) Training of object detectors: The "ground truth" or "label" data y from which an object detector learns must contain localization information ξ^j for each of $j = 1, \dots, N_{\boldsymbol{x}}$ annotated instances on each data point $\boldsymbol{x}$, as well as the associated $N_{\boldsymbol{x}}$ category indices κ^j. Note that we denote labels by the same symbol as the corresponding predicted quantity and omit the hat ($\hat{\cdot}$).

Generically, deep object detectors are trained by stochastic gradient descent or some variant such as AdaGrad [47], or Adam [48] by minimizing an empirical loss

$$\mathcal{L} = \mathcal{L}_\xi + \mathcal{L}_s + \mathcal{L}_p \,. \tag{7}$$

For all object detection frameworks which we consider here (and most architectures, in general) the loss function $\mathcal{L}$ splits up additively into parts punishing localization inaccuracies ($\hat{\xi}$), score ($\hat{s}$) assignment to boxes (assigning large loss to high score for incorrect boxes and low score for correct boxes) and incorrect class probability distribution ($\hat{p}$), respectively. We explicitly give formulas for all utilized loss functions in section B. The trainable weights $\boldsymbol{w}$ of the model are updated in standard gradient descent optimization by

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} \mathcal{L}(\hat{y}(\boldsymbol{x}, \boldsymbol{w}), y) \tag{8}$$

where η is a learning rate factor. We denote by $g(\boldsymbol{x}, \boldsymbol{w}, y) := \nabla_{\boldsymbol{w}} \mathcal{L}(\hat{y}(\boldsymbol{x}, \boldsymbol{w}); y)$ the learning gradient on the data point $(\boldsymbol{x}, y)$.

Appendix B
Implemented Loss Functions

Here, we give a short account of the loss functions implemented in our experiments.

a) YOLOv3: The loss function we used to train YOLOv3 has the following terms:

$$\mathcal{L}_\xi^{\mathrm{YOLOv3}}(\hat{y}, y) = 2 \sum_{a=1}^{N_{\mathrm{out}}} \sum_{t=1}^{N_{\boldsymbol{x}}} \mathbb{I}_{at}^{\mathrm{obj}} \cdot \left[\mathrm{MSE}\left(\begin{pmatrix} \hat{\tau}_{\mathsf{w}}^a \\ \hat{\tau}_{\mathsf{h}}^a \end{pmatrix}, \begin{pmatrix} \tau_{\mathsf{w}}^t \\ \tau_{\mathsf{h}}^t \end{pmatrix} \right) + \mathrm{BCE}\left(\sigma \begin{pmatrix} \hat{\tau}_{\mathsf{x}}^a \\ \hat{\tau}_{\mathsf{y}}^a \end{pmatrix}, \sigma \begin{pmatrix} \tau_{\mathsf{x}}^t \\ \tau_{\mathsf{y}}^t \end{pmatrix} \right) \right], \tag{9}$$

$$\mathcal{L}_s^{\mathrm{YOLOv3}}(\hat{y}, y) = \sum_{a=1}^{N_{\mathrm{out}}} \sum_{t=1}^{N_{\boldsymbol{x}}} \left[\mathbb{I}_{at}^{\mathrm{obj}} \mathrm{BCE}_a\left(\sigma(\hat{\tau}_s), \mathbf{1}_{N_{\mathrm{out}}}\right) + \mathbb{I}_{at}^{\mathrm{noobj}}(y) \mathrm{BCE}_a\left(\sigma(\hat{\tau}_s), \mathbf{0}_{N_{\mathrm{out}}}\right) \right], \tag{10}$$

$$\mathcal{L}_p^{\mathrm{YOLOv3}}(\hat{y}, y) = \sum_{a=1}^{N_{\mathrm{out}}} \sum_{t=1}^{N_{\boldsymbol{x}}} \mathbb{I}_{at}^{\mathrm{obj}} \mathrm{BCE}\left(\sigma(\hat{\tau}_p^a), \sigma(\tau_p^t)\right). \tag{11}$$

Here, the first sum ranges over all N_{out} anchors a and the second sum over the total number N_{gt} of ground truth instances in y. MSE is the usual mean squared error (eq. (14)) for the regression of the bounding box size. As introduced in [2], the raw network outputs (in the notation of theorem 1 this is one component $(\phi_T^c)_{ab}$ of the final feature map ϕ_T, see also section C) for one anchor denoted by

$$\hat{\tau} = (\hat{\tau}_{\mathsf{x}}, \hat{\tau}_{\mathsf{y}}, \hat{\tau}_{\mathsf{w}}, \hat{\tau}_{\mathsf{h}}, \hat{\tau}_s, \hat{\tau}_{p_1}, \dots, \hat{\tau}_{p_C}) \tag{12}$$

are transformed to yield the components of $\hat{y}$:

$$\begin{aligned} \hat{\mathsf{x}} = \ell \cdot \sigma(\hat{\tau}_{\mathsf{x}}) + c_{\mathsf{x}}, \quad \hat{\mathsf{y}} = \ell \cdot \sigma(\hat{\tau}_{\mathsf{y}}) + c_{\mathsf{y}}, \quad \hat{\mathsf{w}} = \pi_{\mathsf{w}} \cdot \mathrm{e}^{\hat{\tau}_{\mathsf{w}}}, \quad \hat{\mathsf{h}} = \pi_{\mathsf{h}} \cdot \mathrm{e}^{\hat{\tau}_{\mathsf{h}}}, \\ \hat{s} = \sigma(\hat{\tau}_s), \quad \hat{p}_j = \sigma(\hat{\tau}_{p_j}). \end{aligned} \tag{13}$$

Here, ℓ is the respective grid cell width/height, σ denotes the sigmoid function, c_{x} and c_{y} are the top left corner position of the respective cell and π_{w} and π_{h} denote the width and height of the bounding box prior (anchor). These relationships can be (at least numerically) inverted to transform ground truth boxes to the scale of $\hat{\tau}$. We denote by $\tau = (\tau_{\mathsf{x}}, \tau_{\mathsf{y}}, \tau_{\mathsf{w}}, \tau_{\mathsf{h}}, \tau_s, \tau_{p_1}, \dots, \tau_{p_C})$ that transformation of the ground truth y which is collected in a feature map γ. Then, we have

$$\mathrm{MSE}(\hat{\tau}, \tau) = \sum_i (\hat{\tau}_i - \tau_i)^2. \tag{14}$$

Whenever summation indices are not clearly specified, we assume from the context that they take on all possible values, e.g. in eq. (9) w and h. Similarly, the binary cross entropy BCE is related to the usual cross entropy loss CE which is commonly used for learning probability distributions:

$$\mathrm{BCE}(p, q) = \sum_i \mathrm{BCE}_i(p, q) = -\sum_i q_i \log(p_i) + (1 - q_i) \log(1 - p_i), \tag{15}$$

$$\mathrm{CE}(p, q) = -\sum_i q_i \log(p_i) \tag{16}$$

where $p, q \in (0, 1)^d$ for some fixed length $d \in \mathbb{N}$. Using binary cross entropy for classification amounts to learning C binary classifiers, in particular the "probabilities" $\hat{p}_j$ are in general not normalized. Note also, that each summand in eq. (10) only has one contribution due to the binary ground truth $\mathbf{1}_{N_{\mathrm{out}}}$, resp. $\mathbf{0}_{N_{\mathrm{out}}}$. The binary cross entropy is also sometimes used for the center location of anchor boxes when the position within each cell is scaled to $(0, 1)$, see $\mathcal{L}_\xi^{\mathrm{YOLOv3}}$.

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

The tensors $\mathbb{I}^{\text{obj}}$ and $\mathbb{I}^{\text{noobj}}$ indicate whether anchor a can be associated to ground truth instance t (obj) or not (noobj) which is determined as in [3] from two thresholds $\varepsilon_+ \geq \varepsilon_- \geq 0$ which are set to 0.5 in our implementation.

Note, that both, $\mathbb{I}^{\text{obj}}$ and $\mathbb{I}^{\text{noobj}}$ only depend on the ground truth y and the fixed anchors, but not on the predictions $\hat{y}$. We express a tensor with entries 1 with the size N as $\mathbf{1}_N$ and a tensor with entries 0 as $\mathbf{0}_N$. The ground truth t one-hot class vector is $\sigma(\tau_p^t) := p^t = (\delta_{i,\kappa_t})_{i=1}^C$.

b) Faster R-CNN: Since Faster R-CNN [3] has a two-stage architecture, there are separate loss contributions for the Region Proposal Network (RPN) and the Region of Interest (RoI) head, the latter of which produces the actual proposals. Formally, writing $\theta_\xi := (\theta_\mathsf{x}, \theta_\mathsf{y}, \theta_\mathsf{w}, \theta_\mathsf{h})$ for the respectively transformed ground truth localization, similarly $\hat{\theta}_\xi$ for the RPN outputs $\hat{y}^{\text{RPN}}$ and $\hat{\theta}_s$ the proposal score output (where $\hat{s}^a = \sigma(\hat{\theta}_s^a)$ is the proposal score):

$$\mathcal{L}_\xi^{\text{RPN}}(\hat{y}^{\text{RPN}}, y) = \frac{1}{|I^+|} \sum_{a=1}^{N_{\text{out}}^{\text{RPN}}} \sum_{t=1}^{N_\text{æ}} I_a^+ \tilde{\mathbb{I}}_{at}^{\text{obj}} \, \text{sm}L_\beta^1 \left(\hat{\theta}_\xi^a, \theta_\xi^t\right), \tag{17}$$

$$\mathcal{L}_s^{\text{RPN}}(\hat{y}^{\text{RPN}}, y) = \sum_{a=1}^{N_{\text{out}}^{\text{RPN}}} \sum_{t=1}^{N_\text{æ}} \left[\mathbb{I}_{at}^{\text{obj}} \text{BCE}_a \left(\sigma(\hat{\theta}_s), \mathbf{1}_{N_{\text{out}}^{\text{RPN}}}\right) + I_a^- \tilde{\mathbb{I}}_{at}^{\text{noobj}} \text{BCE}_a \left(\sigma(\hat{\theta}_s), \mathbf{0}_{N_{\text{out}}^{\text{RPN}}}\right)\right]. \tag{18}$$

The tensors $\tilde{\mathbb{I}}^{\text{obj}}$ and $\tilde{\mathbb{I}}^{\text{noobj}}$ are determined as for YOLOv3 with $\varepsilon_+ = 0.7$ and $\varepsilon_- = 0.3$) but we omit their dependence on y in our notation. Predictions are randomly sampled to contribute to the loss function by the tensors I^+ and I^-, which can be regarded as random variables. The constant batch size b of predictions to enter the RPN loss is a hyperparameter set to 256 in our implementation. We randomly sample $n_+ := \min\{|\tilde{\mathbb{I}}^{\text{obj}}|, b/2\}$ of the $|\tilde{\mathbb{I}}^{\text{obj}}|$ positive anchors (constituting the mask I^+) and $n_- := \min\{|\tilde{\mathbb{I}}^{\text{noobj}}|, b - n_+\}$ negative anchors (I^-). The summation of a ranges over the $N_{\text{out}}^{\text{RPN}}$ outputs of the RPN (in our case, 1000). Proposal regression is done based on the smooth L^1 loss

$$\text{sm}L_\beta^1(\hat{\theta}, \theta) := \sum_i \begin{cases} \frac{1}{2}|\hat{\theta}_i - \theta_i|^2 & \big| \; |\hat{\theta}_i - \theta_i| < \beta \\ |\hat{\theta}_i - \theta_i| - \frac{\beta}{2} & \big| \; |\hat{\theta}_i - \theta_i| \geq \beta \end{cases}, \tag{19}$$

where we use the default parameter choice $\beta = 1/9$. The final prediction $\hat{y}$ of Faster R-CNN is computed from the proposals and RoI results[3] $\hat{\tau}$ in the RoI head:

$$\hat{\mathsf{x}} = \pi_\mathsf{w} \cdot \hat{\tau}_\mathsf{x} + \pi_\mathsf{x}, \quad \hat{\mathsf{y}} = \pi_\mathsf{y} \cdot \hat{\tau}_\mathsf{y} + \pi_\mathsf{y}, \quad \hat{\mathsf{w}} = \pi_\mathsf{w} \cdot \mathrm{e}^{\hat{\tau}_\mathsf{w}}, \quad \hat{\mathsf{h}} = \pi_\mathsf{h} \cdot \mathrm{e}^{\hat{\tau}_\mathsf{h}}, \quad \hat{p} = \Sigma(\hat{\tau}_p), \tag{20}$$

where $\Sigma^i(x) := \mathrm{e}^{x_i} / \sum_j \mathrm{e}^{x_j}$ is the usual softmax function and $\pi := (\pi_\mathsf{x}, \pi_\mathsf{y}, \pi_\mathsf{w}, \pi_\mathsf{h})$ is the respective proposal localization. Denoting with $\tau_\xi := (\tau_\mathsf{x}, \tau_\mathsf{y}, \tau_\mathsf{w}, \tau_\mathsf{h})$ ground truth localization transformed relatively to the respective proposal:

$$\mathcal{L}_\xi^{\text{RoI}}(\hat{y}, y) = \frac{1}{|\mathbb{I}^{\text{obj}}|} \sum_{a=1}^{N_{\text{out}}} \sum_{t=1}^{N_\text{æ}} \mathbb{I}_{at}^{\text{obj}} \, \text{sm}L_\beta^1 \left(\hat{\tau}_\xi^a, \tau_\xi^t\right), \tag{21}$$

$$\mathcal{L}_p^{\text{RoI}}(\hat{y}, y) = \sum_{a=1}^{N_{\text{out}}} \sum_{t=1}^{N_\text{æ}} \left[\mathbb{I}_{at}^{\text{obj}} \, \text{CE}(\Sigma(\hat{\tau}_p^a), p^t) + \mathbb{I}_{at}^{\text{noobj}} \, \text{CE}(\Sigma(\hat{\tau}_{p_0}^a), 1)\right]. \tag{22}$$

Here, $\mathbb{I}^{\text{obj}}$ and $\mathbb{I}^{\text{noobj}}$ are computed with $\varepsilon_+ = \varepsilon_- = 0.5$.

c) RetinaNet: In the RetinaNet [4] architecture, score assignment is part of the classification.

$$\mathcal{L}_\xi^{\text{Ret}}(\hat{y}, y) = \frac{1}{|\mathbb{I}^{\text{obj}}|} \sum_{a=1}^{N_{\text{out}}} \sum_{t=1}^{N_\text{æ}} \mathbb{I}_{at}^{\text{obj}} L^1 \left(\hat{\tau}_\xi^a, \tau_\xi^t\right), \tag{23}$$

$$\begin{aligned} \mathcal{L}_p^{\text{Ret}}(\hat{y}, y) = \frac{1}{|\mathbb{I}^{\text{obj}}|} \sum_{a=1}^{N_{\text{out}}} \sum_{t=1}^{N_\text{æ}} \Bigg[& \mathbb{I}_{at}^{\text{obj}} \sum_{j=1}^{C} \alpha (1 - \sigma(\hat{\tau}_{p_j}^a))^{\gamma_\text{F}} \cdot \text{BCE}_j \left(\sigma(\hat{\tau}_p^a), p^t\right) \\ & + \mathbb{I}_{at}^{\text{noobj}} \, (1 - \alpha) \sigma(\hat{\tau}_{p_0}^a)^{\gamma_\text{F}} \cdot \text{BCE} \left(\sigma(\hat{\tau}_{p_0}^a), 0\right) \Bigg]. \end{aligned} \tag{24}$$

For $\mathbb{I}^{\text{obj}}$ and $\mathbb{I}^{\text{noobj}}$, we use $\varepsilon_+ = 0.5$ and $\varepsilon_- = 0.4$. Regression is based on the absolute loss $L^1(\hat{\tau}, \tau) = \sum_i |\hat{\tau}_i - \tau_i|$ and the classification loss is a formulation of the well-known focal loss with $\alpha = 0.25$ and $\gamma_\text{F} = 2$. Bounding box transformation follows the maps in eq. (20), where π are the respective RetinaNet anchor localizations instead of region proposals. The class-wise scores of the prediction are obtained by $\hat{p}_j = \sigma(\hat{\tau}_{p_j})$, $j = 1, \ldots, C$ as for YOLOv3.

[3]The result $\hat{\tau}$ is similar to eq. (12), but without $\hat{\tau}_s$ where we have instead $C + 1$ classes, with one "background" class. We denote the respective probability by p_0.

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

d) Theoretical loss derivatives: Here, we symbolically compute the loss gradients w.r.t. the network outputs as obtained from our accounts of the loss functions in the previous paragraphs. We do so in order to determine the computational complexity for $D_1\mathcal{L}|_{\phi_T}$ in section C. Note, that for all derivatives of the cross entropy, we can use

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\left[-y\log(\sigma(\tau)) - (1-y)\log(1-\sigma(\tau))\right] = \sigma(\tau) - y. \tag{25}$$

We find for $b = 1, \dots, N_{\text{out}}$ and $r \in \{\mathsf{x}, \mathsf{y}, \mathsf{w}, \mathsf{h}, s, p_1, \dots, p_C\}$

$$\frac{\partial}{\partial \hat{\tau}_r^b}\mathcal{L}_\xi^{\text{YOLOv3}} = 2\sum_{t=1}^{N_{\boldsymbol{x}}} \mathbb{I}_{bt}^{\text{obj}} \left\{ \begin{array}{l|l} \hat{\tau}_r^b - \tau_r^t & r \in \{\mathsf{w}, \mathsf{h}\} \\ \sigma(\hat{\tau}_r^b) - \sigma(\tau_r^t) & r \in \{\mathsf{x}, \mathsf{y}\} \\ 0 & \text{otherwise.} \end{array} \right. , \tag{26}$$

$$\frac{\partial}{\partial \hat{\tau}_r^b}\mathcal{L}_s^{\text{YOLOv3}} = \delta_{rs}\sum_{t=1}^{N_{\boldsymbol{x}}} \left[\mathbb{I}_{bt}^{\text{obj}}(\hat{s}^b - 1) + \mathbb{I}_{bt}^{\text{noobj}}\hat{s}^b\right], \tag{27}$$

$$\frac{\partial}{\partial \hat{\tau}_r^b}\mathcal{L}_p^{\text{YOLOv3}} = \sum_{t=1}^{N_{\boldsymbol{x}}} \mathbb{I}_{bt}^{\text{obj}} \sum_{i=1}^{C} \delta_{rp_i}(\hat{p}_i^b - p_i^t), \tag{28}$$

where δ_{ij} is the Kronecker symbol, i.e., $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. Further, with analogous notation for the output variables of RPN and RoI

$$\frac{\partial}{\partial \hat{\theta}_r^b}\mathcal{L}_\xi^{\text{RPN}} = \frac{1}{|I^+|}\sum_{t=1}^{N_{\boldsymbol{x}}} I_b^+ \tilde{\mathbb{I}}_{bt}^{\text{obj}} \left\{ \begin{array}{l|l} \hat{\theta}_r^b - \theta_r^t & |\hat{\theta}_r^b - \theta_r^t| < \beta \text{ and } r \in \{\mathsf{x}, \mathsf{y}, \mathsf{w}, \mathsf{h}\} \\ \operatorname{sgn}(\hat{\theta}_r^b - \theta_r^t) & |\hat{\theta}_r^b - \theta_r^t| \geq \beta \text{ and } r \in \{\mathsf{x}, \mathsf{y}, \mathsf{w}, \mathsf{h}\} \\ 0 & \text{otherwise} \end{array} \right. , \tag{29}$$

$$\frac{\partial}{\partial \hat{\theta}_r^b}\mathcal{L}_s^{\text{RPN}} = \delta_{rs}\left[I_b^+ \tilde{\mathbb{I}}_{bt}^{\text{obj}}(\hat{s}^b - 1) + I_b^- \tilde{\mathbb{I}}_{bt}^{\text{noobj}}\hat{s}^b\right]. \tag{30}$$

Here, sgn denotes the sign function, which is the derivative of $|\cdot|$ except for the origin. Similarly,

$$\frac{\partial}{\partial \hat{\tau}_r^b}\mathcal{L}_\xi^{\text{RoI}} = \frac{1}{|\mathbb{I}^{\text{obj}}|}\sum_{t=1}^{N_{\boldsymbol{x}}} \mathbb{I}_{bt}^{\text{obj}} \left\{ \begin{array}{l|l} \hat{\tau}_r^b - \tau_r^t & |\hat{\tau}_r^b - \tau_r^t| < \beta \text{ and } r \in \{\mathsf{x}, \mathsf{y}, \mathsf{w}, \mathsf{h}\} \\ \operatorname{sgn}(\hat{\tau}_r^b - \tau_r^t) & |\hat{\tau}_r^b - \tau_r^t| \geq \beta \text{ and } r \in \{\mathsf{x}, \mathsf{y}, \mathsf{w}, \mathsf{h}\} \\ 0 & \text{otherwise} \end{array} \right. , \tag{31}$$

$$\frac{\partial}{\partial \hat{\tau}_r^b}\mathcal{L}_p^{\text{RoI}} = -\sum_{t=1}^{N_{\boldsymbol{x}}} \left[\mathbb{I}_{bt}^{\text{obj}} \sum_{j=1}^{C} p_j^t \left(\delta_{p_j r} - \sum_{k=0}^{C} \delta_{p_k r} \Sigma^k(\hat{\tau}_p^b)\right) + \mathbb{I}_{bt}^{\text{noobj}} \left(\delta_{p_0 r} - \sum_{k=0}^{C} \delta_{p_k r} \Sigma^k(\hat{\tau}_p^b)\right)\right]. \tag{32}$$

Note that the inner sum over j only has at most one term due to $\delta_{p_j r}$. With $\sigma'(\tau) = \sigma(\tau)(1 - \sigma(\tau))$, we finally find for RetinaNet

$$\frac{\partial}{\partial \hat{\tau}_r^b}\mathcal{L}_\xi^{\text{Ret}} = \frac{1}{|\mathbb{I}^{\text{obj}}|}\sum_{t=1}^{N_{\boldsymbol{x}}} \mathbb{I}_{bt}^{\text{obj}} \left\{ \begin{array}{l|l} \operatorname{sgn}(\hat{\tau}_r^b - \tau_r^t) & r \in \{\mathsf{x}, \mathsf{y}, \mathsf{w}, \mathsf{h}\} \\ 0 & \text{otherwise} \end{array} \right. , \tag{33}$$

$$\begin{aligned} \frac{\partial}{\partial \hat{\tau}_r^b}\mathcal{L}_p^{\text{Ret}} = \frac{1}{|\mathbb{I}^{\text{obj}}|}\sum_{t=1}^{N_{\boldsymbol{x}}} \Bigg[& \mathbb{I}_{bt}^{\text{obj}} \sum_{j=1}^{C} \delta_{p_j r}\alpha(1 - \sigma(\hat{\tau}_{p_j}^b))^{\gamma_{\text{F}}}[-\gamma_{\text{F}}\sigma(\hat{\tau}_{p_j}^b)\text{BCE}_j(\sigma(\hat{\tau}_p^b), p^t) + \sigma(\hat{\tau}_{p_j}^b) - 1] \\ & + \mathbb{I}_{bt}^{\text{noobj}}\delta_{p_0 r}(1-\alpha)\sigma(\hat{\tau}_{p_0}^b)^{\gamma_{\text{F}}}[-\gamma_{\text{F}}(1 - \sigma(\hat{\tau}_{p_0}^b))\log(1 - \sigma(\hat{\tau}_{p_0}^b)) + \sigma(\hat{\tau}_{p_0}^b)]\Bigg]. \end{aligned} \tag{34}$$

APPENDIX C
COMPUTATIONAL COMPLEXITY

In this section, we discuss the details of the setting in which theorem 1 was formulated an give a proof for the statements made there. The gradients for our uncertainty metrics are usually computed via backpropagation only for a few network layers. Therefore, we restrict ourselves to the setting of fully convolutional neural networks.

a) Setting: As in [38, Chapter 20.6], we regard a (convolutional) neural network as a graph of feature maps with vertices $V = \bigsqcup_{t=0}^{T} V_t$ arranged in layers V_t. For our consideration it will suffice to regard them as sequentially ordered. We denote $[n] := \{1, \dots, n\}$ for $n \in \mathbb{N}$. Each layer V_t contains a set number $k_t := |V_t|$ of feature map activations ("channels") $\phi_t^c \in \mathbb{R}^{h_t \times w_t}$, $c \in [k_t]$. We denote the activation of V_t by $\phi_t = (\phi_t^1, \dots, \phi_t^{k_t})$. The activation $\phi_{t+1} \in (\mathbb{R}^{h_{t+1} \times w_{t+1}})^{k_{t+1}}$ is obtained from ϕ_t by convolutions. We have $k_t \times k_{t+1}$ quadratic filter matrices

$$(K_{t+1})_c^d \in \mathbb{R}^{(2s_t+1)\times(2s_t+1)}, \quad c \in [k_t]; \quad d \in [k_{t+1}], \tag{35}$$

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

where s_t is a (usually small) natural number, the spatial extent of the filter. Also, we have respectively k_{t+1} biases $b_{t+1}^d \in \mathbb{R}$, $d \in [k_{t+1}]$. The convolution (actually in most implementations, the cross correlation) of $K \in \mathbb{R}^{(2s+1)\times(2s+1)}$ and $\phi \in \mathbb{R}^{h\times w}$ is defined as

$$(K * \phi)_{ab} := \sum_{m,n=-s}^{s} K_{s+1+p,s+1+q}\phi_{a+p,b+q}, \quad a = 1,\dots,h, \quad b = 1,\dots,w \tag{36}$$

This is, strictly speaking, only correct for convolutions with "stride" 1, although a closed form can be given for the more general case. For our goals, we will use stride 1 to upper bound the FLOPs which comes with the simplification that the feature maps' sizes are conserved. We then define

$$\psi_{t+1}^d = \sum_{c=1}^{k_t} (K_{t+1})_c^d * \phi_t^c + b_{t+1}^d \mathbf{1}_{h_t\times w_t}, \quad d \in [k_{t+1}]. \tag{37}$$

Finally, we apply activation functions $\alpha_t : \mathbb{R} \to \mathbb{R}$ to each entry to obtain $\phi_{t+1} = \alpha_{t+1}(\psi_{t+1})$. In practice, α_t is usually a ReLU activation, i.e., $\alpha_t(x) = \max\{x, 0\}$ or a slight modification (e.g. leaky ReLU) of it and we will treat the computational complexity of this operation later. We can then determine the computational expense of computing ψ_{t+1} from ϕ_t. In the following, we will be interested in the linear convolution action

$$C^{K_t} : \mathbb{R}^{k_{t-1}\times h_{t-1}\times w_{t-1}} \to \mathbb{R}^{k_t\times h_t\times w_t}, \qquad (C^{K_t}\phi_{t-1})_{ab}^d := \left(\sum_{c=1}^{k_t} (K_t)_c^d * \phi_t^c\right)_{ab}, \tag{38}$$

where $d \in [k_t]$, $a \in [h_t]$ and $b \in [w_t]$. Note that C^K is also linear in K_t. On the last layer feature map ϕ_T we define the loss function $\mathcal{L} : (\phi_T, \gamma) \mapsto \mathcal{L}(\phi_T, \gamma) \in \mathbb{R}$. Here, γ stands for the ground truth[4] transformed to feature map size $\mathbb{R}^{h_T\times w_T\times k_T}$. In order to make dependencies explicit, define the loss of the subnet starting at layer t by ℓ_t, i.e.,

$$\ell_T(\phi_T) := \mathcal{L}(\phi_T, \gamma), \qquad \ell_{t-1}(\phi_{t-1}) := \ell_t(\alpha_t(\psi_t)). \tag{39}$$

Straight-forward calculations yield

$$\begin{aligned} \nabla_{K_T}\mathcal{L} =& \nabla_{K_T}\left(\ell_T \circ \alpha_T \circ \psi_T(K_T)\right) \\ =& D_1\mathcal{L}|_{\phi_T} \cdot D\alpha_T|_{\psi_T} \cdot \nabla_{K_T}\psi_T \end{aligned} \tag{40}$$

$$\begin{aligned} \nabla_{K_{T-1}}\mathcal{L} =& \nabla_{K_{T-1}}\left(\ell_T \circ \alpha_T \circ \psi_T \circ \alpha_{T-1} \circ \psi_{T-1}(K_{T-1})\right) \\ =& D_1\mathcal{L}|_{\phi_T} \cdot D\alpha_T|_{\psi_T} \cdot C^{K_T} \cdot D\alpha_{T-1}|_{\psi_{T-1}} \cdot \nabla_{K_{T-1}}\psi_{T-1}. \end{aligned} \tag{41}$$

Here, D denotes the total derivative (D_1 for the first variable, resp.) and we have used the linearity of C^{K_T}. Note, that in section III, we omitted the terms $D\alpha_T|_{\psi_T}$ and $D\alpha_{T-1}|_{\psi_{T-1}}$. We will come back to them later in the discussion. For the gradient metrics we present in this paper, each $\hat{y}^j$ for which we compute gradients receives a binary mask μ^j such that $\mu^j \cdot \phi_T$ are the feature map representations of candidate boxes for $\hat{y}^j$ (see section A). The scalar loss function then becomes $\mathcal{L}(\mu_j\phi_T, \gamma^j)$ for the purposes of computing gradient uncertainty, where γ^j is $\hat{y}^j$ in feature map representation. We address next, how this masking influences eq. (40), eq. (41) and the FLOP count of our method.

b) Computing the mask: The complexity of determining μ^j (i.e., finding $\mathrm{cand}[\hat{y}^j]$) is the complexity of computing all mutual IoU values between $\hat{y}^j$ and the $n_T := h_T \cdot w_T \cdot k_T$ other predicted boxes. Computing the IoU of a box $b_1 = (x_1^{\min}, y_1^{\min}, x_1^{\max}, y_1^{\max})$ and $b_2 = (x_2^{\min}, y_2^{\min}, x_2^{\max}, y_2^{\max})$ can be done in a few steps with an efficient method exploiting the fact that:

$$\begin{aligned} U =& A_1 + A_2 - I, \qquad IoU = I/U, \end{aligned} \tag{42}$$

where the computation of the intersection area I and the individual areas A_1 and A_2 can each be done in 3 FLOP, resulting in 12 FLOP per pair of boxes. Note that different localization constellations of b_1 and b_2 may result in slightly varying formulas for the computation of I but the constellation can be easily detemined by binary checks which we ignore computationally. Also, the additional check for the class and sufficient score will be ignored, so we have $12n_T$ FLOP per mask μ^j. Inserting the binary mask[5] μ^j in eq. (40) and eq. (41) leads to the replacement of $D_1\mathcal{L}|_{\phi_T} \cdot D\alpha_T|_{\psi_T}$ by $D\mathcal{L}^j := D_1\mathcal{L}(\cdot, \gamma^j)|_{\mu^j\phi_T} \cdot \mu^j \cdot D\alpha_T|_{\psi_T}$ for each relevant box $\hat{y}^j$.

In table V we have listed upper bounds on the number of FLOP and elementary function evaluations performed for the computation of $D\mathcal{L}^j$ for the investigated loss functions. The numbers were obtained from the explicit partial derivatives computed in section B. In principle, those formulas allow for every possible choice of $b \in [N_{\mathrm{out}}]$ which is why all counts are proportional to it. Practically, however, at most the $|\mu^j|$ candidate boxes are relevant which need to be identified additionally as foreground or background for $\hat{y}^j$ in a separate step involving an IoU computation between $\hat{y}^j$ and the respective anchor. The

[4]The transformations are listed in section B for the entries τ of ϕ_T.

[5]See section III. The mask μ^j selects the feature map representation of $\mathrm{cand}[\hat{y}^j]$ out of ϕ_T.

TABLE V
UPPER BOUNDS ON FLOP AND ELEMENTARY FUNCTION EVALUATIONS PERFORMED DURING THE COMPUTATION OF $D\mathcal{L}^j$ (ALL CONTRIBUTIONS) AND MC DROPOUT POST PROCESSING (PP) FOR N_{samp} DROPOUT SAMPLES.

	YOLOv3	Faster R-CNN	RetinaNet
# FLOP $D\mathcal{L}^j$	$(9+C)N_{\text{out}}$	$10N_{\text{out}}^{\text{RPN}} + (2+2C)N_{\text{out}}$	$(18+11C)N_{\text{out}}$
# FLOP MC PP	$8N_{\text{out}}N_{\text{samp}}$	$(9+2C)N_{\text{out}}N_{\text{samp}}$	$8N_{\text{out}}N_{\text{samp}}$
# evaluations $D\mathcal{L}^j$	0	0	$2(1+C)N_{\text{out}}$
# evaluations MC PP	$(5+C)N_{\text{out}}N_{\text{samp}}$	$(3+C)N_{\text{out}}N_{\text{samp}}$	$(3+C)N_{\text{out}}N_{\text{samp}}$

total count of candidate boxes in practice is on average not larger than ~ 30. When evaluating the formulas from section B note, that there is only one ground truth box per gradient and we assume here, that one full forward pass has already been performed such that the majority of the appearing evaluations of elementary functions (sigmoids, exponentials, etc.) have been computed beforehand. This is not the case for the RetinaNet classification loss (34). In table V we also list the additional post-processing cost for the output transformations (see section B, eqs. (13) and (20)) required in the MC dropout samples ("MC PP"). The latter are also proportional to N_{out}, but also to the number N_{samp} of samples.

c) Proof of theorem 1: Before we begin the proof, we first re-state the claims of theorem 1.

Theorem 1. *In the setting explained above, the number of FLOP required to compute the gradient $\nabla_{K_T}\mathcal{L}(\mu^j\phi_T(K_T),\gamma^j)$ is $\mathcal{O}(k_T hw + k_T k_{T-1}(2s_T+1)^4)$. Simlarly, for earlier layers, i.e., $\nabla_{K_t}\mathcal{L}(\mu^j\phi_T(K_t),\gamma^j)$, we have $\mathcal{O}(k_{t+1}k_t + k_t k_{t-1})$, provided that we have previously computed the gradient for the consecutive layer $t+1$. Performing MC dropout with dropout on ϕ_{T-1} requires $\mathcal{O}(k_T k_{T-1} hw)$ FLOP per dropout sample.*

Our implementations exclusively use stride 1 convolutions for the layers indicated in section IV, so $w_T = w_{T-1} = w_{T-2} =: w$, resp. $h_T = h_{T-1} = h_{T-2} =: h$. As before, we denote $n_t := hwk_t$, and regard $D\mathcal{L}^j$ as a $1 \times n_T$ matrix. Next, regard the matrix-vector multiplication to be performed in eq. (40). Since for all $t \in [T]$ we have that ψ_t is linear in K_t, we regard $\nabla_{K_t}\psi_t$ as a matrix acting on the filter space $\mathbb{R}^{k_{t-1}\times k_t\times(2s_t+1)^2}$. For $d \in [k_t]$, ψ_t^d only depends on K_t^d (see eq. (37)), so $\nabla\psi_t$ only has at most $k_{t-1}\cdot(2s_t+1)^2\cdot n_t$ non-vanishing entries. Therefore, regard it as a $n_t \times (k_{t-1}(2s_t+1)^2)$ matrix. We will now show that this matrix has $k_t(2s_t+1)^2$-sparse columns.

Let $c \in [k_t]$, $d \in [k_{t-1}]$, $p,q \in \{-s_t,\dots,s_t\}$, $a \in [h_t]$ and $b \in [w_t]$. One easily sees from eqs. (36) and (37) that

$$\frac{\partial}{\partial((K_t)_c^d)_{pq}}(\psi_t)_{ab}^d = (\phi_{t-1})_{a+p-s_t-1,b+q-s_t-1}^c, \tag{43}$$

where ϕ_{t-1}^c is considered to vanish for $a+p-s_t-1 \notin [h_t]$ and $b+q-s_t-1 \notin [w_t]$. Consistency with the definition of p and q requires that both the conditions

$$1 < a \le 2s_t+2, \qquad 1 < b \le 2s_t+2 \tag{44}$$

are satisfied, which means that $(\nabla_{K_t}\psi_t)^d$ can only have $k_t(2s_t+1)^2$ non-zero entries. Appealing to sparsity $\nabla_{K_T}\psi_T$ in eq. (40) is then, effectively, a $(k_{T-1}\cdot(2s_T+1)^2)\times(k_T\cdot(2s_T+1)^2)$ matrix, resulting in a FLOP count of

$$[2\cdot k_T(2s_T+1)^2 - 1]\cdot[k_{T-1}\cdot(2s_T+1)^2] \tag{45}$$

for the multiplication $D\mathcal{L}^j\cdot\nabla_{K_T}\psi_T$ giving the claimed complexity considering that the computation of μ^j is $\mathcal{O}(k_T hw)$.

Next, we investigate the multiplication in eq. (41), in particular the multiplication $D\mathcal{L}^j\cdot C^{K_T}$ as the same sparsity argument applies to $\nabla_{K_{T-1}}\psi_{T-1}$. First, for $t \in [T]$, regard C^{K_t} as a $n_t \times n_{t-1}$ matrix acting on a feature map $\phi \in \mathbb{R}^{n_{t-1}}$ from the left via

$$\left(C^{K_t}\phi\right)_{ab}^d = \sum_{c=1}^{k_{t-1}}\left[(K_t)_c^d * \phi^c\right]_{ab} = \sum_{c=1}^{k_{t-1}}\sum_{m,n=-s_t}^{s_t}[(K_t)_c^d]_{s_t+1+m,s_t+1+n}(\phi^c)_{a+m,b+n}, \tag{46}$$

where $d \in [k_t]$, $b \in [w_t]$ and $a \in [h_t]$ indicate one particular row in the matrix representation of C^{K_t}. From this, we see the sparsity of C^{K_t}, namely the multiplication result of row (d,a,b) acts on at most $k_{t-1}\cdot(2s_t+1)^2$ components of ϕ_{t-1} (i.e., $k_{t-1}(2s_t+1)^2$-sparsity of the rows). Conversely, we also see that at most $k_t\cdot(2s_t+1)^2$ convolution products $(C^{K_t}\phi)_{ab}^d$ have a dependency on one particular feature map pixel $(\phi^c)_{\tilde{a}\tilde{b}}$ (i.e., $k_t(2s_t+1)^2$-sparsity of the columns). Now, let $t \in [T-1]$ and assume that we have already computed the gradient

$$\nabla_{K_{t+1}}\mathcal{L} = \nabla_{K_{t+1}}\ell_{t+1}(\phi_{t+1}(K_{t+1})) = D\ell_{t+1}|_{\phi_{t+1}}\cdot\alpha_{t+1}|_{\psi_{t+1}}\cdot\nabla_{K_{t+1}}\psi_{t+1}, \tag{47}$$

then by backpropagation, i.e., eq. (39), we obtain

$$\nabla_{K_t}\mathcal{L} = \nabla_{K_t}[\ell_{t+1}\circ\alpha_{t+1}\circ\psi_{t+1}(\phi_t(K_t))] = D\ell_{t+1}|_{\phi_{t+1}}\cdot\alpha_{t+1}|_{\psi_{t+1}}\cdot C^{K_{t+1}}\cdot D\alpha_t|_{\psi_t}\cdot\nabla_{K_t}\psi_t. \tag{48}$$

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

Here, the first two factors have already been computed, hence we obtain a FLOP count for subsequently computing $\nabla_{K_t}\mathcal{L}$ of

$$[2 \cdot k_{t+1}(2s_{t+1}+1)^2 - 1] \cdot [k_t(2s_t+1)^2] + [2 \cdot k_t(2s_t+1)^2 - 1] \cdot [k_{t-1}(2s_t+1)^2] \tag{49}$$

via the backpropagation step from $\nabla_{K_{t+1}}\mathcal{L}$. The claim in theorem 1 addressing eq. (41), follows for $t = T-1$ in eq. (49).

Finally, we address the computational complexity for MC dropout with dropout on ϕ_{T-1}. Again, we ignore the cost of dropout itself as it is random binary masking together with a respective up-scaling/multiplication of the non-masked entries by a constant. The cost stated in theorem 1 results from the residual forward pass $\phi_{T-1} \mapsto \phi_T = \alpha_T(C^{K_T} \cdot \phi_{T-1} + b_T)$ where we now apply previous results. Obtaining all n_T entries in the resulting sample feature map requires a total FLOP count of

$$2n_T k_{T-1}(2s_T+1)^2 - 1 + n_T \tag{50}$$

as claimed, where we have considered the sparsity of C^{K_T}. The last term results from the bias addition.

d) Discussion: A large part of the FLOP required to compute gradient metrics results from the computation of the masks μ^j and the term $D\mathcal{L}^j$ for each relevant predicted box. In table V we have treated the latter separately and found that, although the counts listed for $D\mathcal{L}^j$ apply to each separate box, MC dropout post-processing comes with considerable computational complexity as well. In that regard, we have similar costs for gradient metrics and MC dropout. Note in particular, that computing $D\mathcal{L}^j$ requires no new evaluation of elementary functions, as opposed to MC dropout. Once $D\mathcal{L}^j$ is computed for $\hat{y}^j$, the last layer gradient can be computed in $\mathcal{O}(k_T k_{T-1})$ and every further gradient for layer V_t in $\mathcal{O}(k_{t+1}k_t + k_t k_{t-1})$. Each dropout sample results in $\mathcal{O}(n_T k_{T-1})$ with dropout on ϕ_{T-1}. Performing dropout any earlier results in additional full convolution forward passes which also come with considerable computational costs. We note that MC dropout uncertainty can be computed in parallel with all N_{samp} forward passes being performed simultaneously. Gradient uncertainty metrics, in contrast, require one full forward pass for the individual gradients $\nabla_t\mathcal{L}(\mu^j\phi_T(K_t), \gamma^j)$ to be computed. Therefore, gradient uncertainty metrics experience a slight computational latency as compared to MC dropout. We argue that in principle, all following steps (computation of μ^j and $\nabla_t\mathcal{L}(\mu^j\phi_T(K_t), \gamma^j)$) can be implemented to run in parallel as no sequential order of computations is required. We have not addressed the computations of mapping the gradients to scalars from eq. (3) which are roughly comparable to the cost of computing the sample std for MC dropout, especially once the sparsity of $D\mathcal{L}^j$ has been determined in the computation of $\nabla_{K_T}\mathcal{L}$. The latter also brings a significant reduction in FLOP (from n_T to $|\mu^j|$) which cannot be estimated more sharply, however. Since $D\mathcal{L}^j$ is sparse, multiplication from the right with $D\alpha_T|_{\psi_T}$ in eqs. (40) and (41) for a leaky ReLU activation only leads to lower-order terms. The same terms were also omitted before in determining the computational complexity of MC dropout. Also, for this consideration, we regard the fully connected layers used for bounding box regression and classification in the Faster R-CNN RoI head as 1×1 convolutions to stay in the setting presented here.

APPENDIX D
CALIBRATION

Generally, "calibration" methods aim at rectifying scores as confidences in the sense of section I such that the calibrated scores reflect the conditional frequency of true predictions. For example, out of 100 predictions with a confidence of 0.3, around 30 should be correct.

Confidence calibration methods have been applied to object detection in [49] where temperature scaling was found to improve calibration. In addition to considering the expected calibration error (ECE) and the maximum calibration error (MCE) [50], the authors of [49] argue that in object detection, it is important that confidences are calibrated irrespective of how many examples fall into a bin. Therefore, they introduced the average calibration error (ACE) as a new calibration metric which is insensitive to the bin counts. The authors of [51] introduce natural extensions to localization-dependent calibration methods and a localization-dependent metric to measure calibration for different image regions.

In Sec. IV, we evaluated the calibration of meta classifiers in terms of the maximum (MCE, [50]) and average (ACE, [49]) calibration error which we define here. We sort the examples into bins β_i, $i = 1, \dots, B$ of a fixed width (in our case 0.1) according to their confidence. For each bin β_i, we compute

$$\mathrm{acc}_i = \frac{\mathrm{TP}_i}{|\beta_i|}, \qquad \mathrm{conf}_i = \frac{1}{|\beta_i|}\sum_{j=1}^{|\beta_i|}\hat{c}_i \tag{51}$$

where $|\beta_i|$ denotes the number of examples in β_i and $\hat{c}_i$ is the respective confidence, i.e., the network's score or a meta classification probability. TP_i denotes the number of correctly classified in β_i. The mentioned metrics are then

$$MCE = \max_{i=1,\dots,B}|\mathrm{acc}_i - \mathrm{conf}_i|, \qquad ACE = \frac{1}{B}\sum_{i=1}^{B}|\mathrm{acc}_i - \mathrm{conf}_i|, \tag{52}$$

$$ECE = \frac{1}{B}\sum_{i=1}^{B}\frac{B}{|\beta_i|}|\mathrm{acc}_i - \mathrm{conf}_i|. \tag{53}$$

Table VI also shows the expected (ECE, [50]) calibration error which was argued in [49] to be biased toward bins with large amounts of examples. ECE is, thus, less informative for safety-critical investigations.

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

Preprint – Preprint – Preprint – Preprint – Preprint

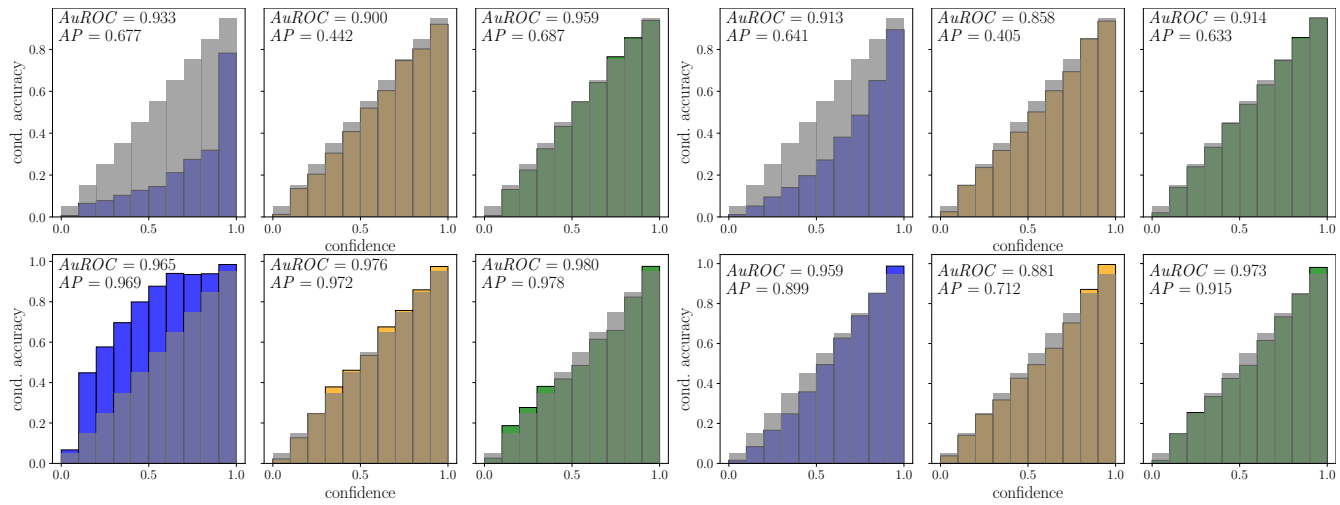

Fig. 5. Reliability diagrams of score, resp. meta classifiers for *top left*: (Faster R-CNN, VOC); *top right*: (Faster R-CNN, COCO); *bottom left*: (YOLOv3, KITTI); *bottom right*: (RetinaNet, KITTI). Each panel shows the score (*left*), MC dropout uncertainty (*center*) and gradient uncertainty G (*right*).

APPENDIX E
FURTHER NUMERICAL RESULTS

a) Calibration of meta classifiers: For sake of completeness, we list in table VI the calibration metrics ECE, MCE and ACE defined in section D for score and meta classifiers for all object detectors on all three datasets investigated in section IV. This can be seen as an extension of table III. All calibration metrics are in line with the results from section IV with meta classifiers being always better calibrated than the score by at least half an order of magnitude in any calibration metric. The ECE metric is comparatively small for all meta classifiers and, therefore, insensitive and harder to interpret than MCE and ACE. As was argued in [49], the former is also less informative as bin-wise accuracy is weighted with the bin counts. In table VI we can see a weakly increasing trend of calibration errors in the meta classifiers due to overfitting on the increasing number of co-variables. Similarly to the reliability histograms in fig. 1 for Faster R-CNN on KITTI, we show analogous diagrams in fig. 5 for Faster R-CNN on VOC and COCO, as well as for YOLOv3 and RetinaNet on the KITTI dataset. Meta classifiers are well-calibrated across the board. For Faster R-CNN (top row) we see the typical network score overconfidence by the discrepancy to the diagonal towards low accuracy. In contrast, on the bottom left (YOLOv3, KITTI) we see strong underconfidence in the network score which is still leading to large calibration errors and is, as mentoned in section I, arguably even more dangerous than overconfidence. The bottom right panel (RetinaNet, KITTI) only shows small overconfidence for the score in the range between 0.1 and 0.5 which is also reflected in the comparably less severe calibration errors computed in table VI.

b) Performance of meta classifiers: The meta classifiers for the same combinations of uncertainty metrics as in table II for Faster R-CNN and RetinaNet can be found in table VII. The findings are similar to the analogous table for YOLOv3 with the combination models G+MD or G+MD+MC having the best overall predictive power. Note that there is some overfitting in G+MD+MC in particular on Faster R-CNN where we compute the largest number of gradients (see table I). In those cases there is no additional boost from MC dropout. We find that MC dropout produces large standard deviations in Faster R-CNN and RetinaNet for all boxes except for those with very high score. The respective entries in table VII suggest that MC dropout uncertainty metrics have low predictive power in those cases. They, therefore, contain little to no valuable additional uncertainty information. Our gradient-based model G is roughly on par with MD, outperforming the latter in terms of $AuROC$ but trailing slightly in terms of AP. For Faster R-CNN, G alone reaches an $AuROC$ of 0.99 on the KITTI dataset. Figure 6 shows violin plots for confidence samples of the cross-validated results in table VII for Faster R-CNN on the KITTI dataset. The violins for Score (left) and G (right) look similar while the violin of MC (center) is less concentrated, reflecting the lower $AuROC$ found in table VII. We see that combining MD with G consistently leads to significant performance boosts, indicating mutual non-redundancy as already seen in section IV.

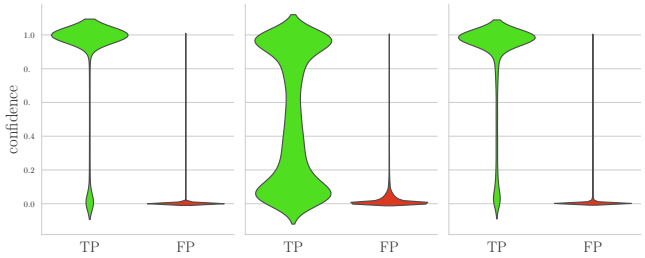

Fig. 6. Confidence violin plots divided into TP and FP for Score (left), MC (center) and G (right). Model: Faster R-CNN, dataset KITTI evaluation split.

TABLE VI
EXPECTED (ECE, [50]), MAXIMUM (MCE, [50]) AND AVERAGE (ACE, [49]) CALIBRATION ERRORS PER CONFIDENCE MODEL OVER 10-FOLD CV (std IN PARENTHESES).

	Pascal VOC			COCO			KITTI		
YOLOv3	ECE	MCE	ACE	ECE	MCE	ACE	ECE	MCE	ACE
Score	0.053	0.237	0.108	0.033	0.065	0.041	0.068	0.348	0.227
MC	0.006 (0.000)	0.035 (0.006)	0.013 (0.001)	0.005 (0.000)	0.024 (0.004)	0.011 (0.001)	**0.005 (0.001)**	**0.040 (0.017)**	**0.014 (0.004)**
$\|\cdot\|_2$	0.005 (0.001)	0.033 (0.006)	0.014 (0.002)	**0.002 (0.001)**	**0.013 (0.004)**	**0.004 (0.001)**	0.008 (0.001)	0.046 (0.011)	0.019 (0.003)
$\|\cdot\|_1$+$\|\cdot\|_2$	**0.004 (0.000)**	**0.031 (0.005)**	**0.012 (0.001)**	0.003 (0.000)	**0.013 (0.003)**	0.006 (0.001)	0.009 (0.001)	0.058 (0.023)	0.022 (0.006)
G	0.006 (0.001)	0.051 (0.011)	0.021 (0.003)	0.004 (0.000)	0.036 (0.006)	0.014 (0.001)	0.012 (0.001)	0.086 (0.017)	0.038 (0.004)
MD	0.006 (0.000)	0.048 (0.014)	0.018 (0.002)	0.006 (0.000)	0.033 (0.005)	0.014 (0.001)	0.011 (0.001)	0.078 (0.014)	0.028 (0.004)
MD+MC	0.007 (0.000)	0.054 (0.007)	0.021 (0.002)	0.005 (0.000)	0.033 (0.001)	0.014 (0.001)	0.012 (0.001)	0.065 (0.017)	0.028 (0.004)
G+MC	0.008 (0.001)	0.067 (0.010)	0.029 (0.004)	0.006 (0.000)	0.043 (0.006)	0.016 (0.001)	0.014 (0.001)	0.094 (0.015)	0.041 (0.005)
G+MD	0.008 (0.000)	0.061 (0.008)	0.025 (0.002)	0.007 (0.000)	0.044 (0.005)	0.020 (0.001)	0.014 (0.001)	0.095 (0.025)	0.043 (0.007)
G+MD+MC	0.008 (0.001)	0.062 (0.012)	0.027 (0.003)	0.006 (0.000)	0.041 (0.004)	0.018 (0.001)	0.015 (0.001)	0.088 (0.022)	0.044 (0.004)
Faster R-CNN									
Score	0.016	0.534	0.287	0.057	0.275	0.179	0.007	0.376	0.128
MC	0.001 (0.000)	0.064 (0.009)	0.032 (0.004)	0.002 (0.000)	0.062 (0.013)	0.026 (0.005)	0.001 (0.000)	**0.045 (0.016)**	**0.015 (0.003)**
$\|\cdot\|_2$	**0.000 (0.000)**	0.026 (0.004)	0.009 (0.002)	**0.001 (0.000)**	**0.014 (0.005)**	**0.004 (0.001)**	**0.000 (0.000)**	0.051 (0.013)	0.019 (0.004)
$\|\cdot\|_1$+$\|\cdot\|_2$	**0.000 (0.000)**	**0.025 (0.006)**	**0.008 (0.001)**	**0.001 (0.000)**	0.019 (0.004)	0.007 (0.001)	**0.000 (0.000)**	0.048 (0.015)	0.016 (0.003)
G	0.001 (0.000)	0.035 (0.009)	0.013 (0.002)	**0.001 (0.000)**	0.016 (0.002)	0.007 (0.001)	0.001 (0.000)	0.068 (0.020)	0.022 (0.005)
MD	**0.000 (0.000)**	0.028 (0.010)	0.011 (0.003)	**0.001 (0.000)**	0.015 (0.003)	0.005 (0.001)	0.001 (0.000)	0.067 (0.018)	0.020 (0.004)
MD+MC	**0.000 (0.000)**	0.028 (0.009)	0.009 (0.002)	**0.001 (0.000)**	0.016 (0.002)	0.005 (0.001)	0.001 (0.000)	0.055 (0.020)	0.018 (0.005)
G+MC	0.001 (0.000)	0.044 (0.008)	0.017 (0.002)	**0.001 (0.000)**	0.020 (0.005)	0.007 (0.001)	0.001 (0.000)	0.087 (0.013)	0.026 (0.004)
G+MD	0.001 (0.000)	0.041 (0.010)	0.017 (0.002)	0.002 (0.000)	0.023 (0.005)	0.009 (0.001)	0.001 (0.000)	0.072 (0.015)	0.027 (0.003)
G+MD+MC	0.001 (0.000)	0.045 (0.006)	0.019 (0.002)	0.002 (0.000)	0.022 (0.003)	0.009 (0.001)	0.001 (0.000)	0.069 (0.018)	0.024 (0.003)
RetinaNet									
Score	0.068	0.212	0.123	0.089	0.192	0.106	0.027	0.097	0.043
MC	0.004 (0.000)	0.112 (0.031)	0.041 (0.007)	**0.001 (0.000)**	0.066 (0.014)	0.021 (0.004)	0.004 (0.001)	0.051 (0.012)	0.017 (0.004)
$\|\cdot\|_2$	**0.002 (0.000)**	**0.021 (0.005)**	**0.007 (0.001)**	0.003 (0.000)	**0.018 (0.003)**	**0.006 (0.001)**	**0.002 (0.000)**	**0.042 (0.008)**	**0.012 (0.002)**
$\|\cdot\|_1$+$\|\cdot\|_2$	**0.002 (0.000)**	0.028 (0.009)	0.009 (0.002)	0.004 (0.000)	0.022 (0.004)	0.009 (0.001)	0.003 (0.000)	0.049 (0.010)	0.014 (0.002)
G	0.003 (0.000)	0.044 (0.009)	0.014 (0.001)	0.005 (0.000)	0.031 (0.006)	0.012 (0.001)	0.005 (0.001)	0.060 (0.013)	0.022 (0.004)
MD	0.003 (0.000)	0.031 (0.008)	0.011 (0.002)	0.005 (0.001)	0.022 (0.004)	0.009 (0.001)	0.003 (0.000)	0.044 (0.006)	0.016 (0.002)
MD+MC	0.003 (0.000)	0.031 (0.006)	0.013 (0.001)	0.005 (0.000)	0.022 (0.005)	0.010 (0.001)	0.004 (0.000)	0.045 (0.010)	0.016 (0.002)
G+MC	0.004 (0.000)	0.054 (0.010)	0.020 (0.002)	0.006 (0.000)	0.026 (0.003)	0.012 (0.001)	0.005 (0.001)	0.064 (0.017)	0.024 (0.004)
G+MD	0.005 (0.000)	0.064 (0.008)	0.024 (0.002)	0.007 (0.001)	0.032 (0.004)	0.015 (0.001)	0.006 (0.000)	0.070 (0.014)	0.028 (0.003)
G+MD+MC	0.005 (0.000)	0.056 (0.010)	0.022 (0.002)	0.007 (0.000)	0.032 (0.007)	0.015 (0.001)	0.006 (0.001)	0.064 (0.010)	0.027 (0.004)

c) Performance of meta regression: As for the meta classifiers we also gather the complementary results for Faster R-CNN and RetinaNet in table VIII. There is a trend similar to the meta classification performance in that we have low predictive power in MC dropout in all instances and we also see the corresponding overfitting between G+MD and G+MD+MC. Here, we particularly see the predictive power of G which is outperforming MD across the board. Also, all combination models involving G (i.e., G+MC, G+MD, G+MD+MC) see significant improvement compared to their counter part without G. In addition to the scatter plots in fig. 4 for YOLOv3 on the KITTI evaluation split, we also show scatter plots in fig. 8 for YOLOv3 on VOC and COCO, as well as for Faster R-CNN and RetinaNet on KITTI. For Faster R-CNN and RetinaNet, we show G+MD instead of G+MD+MC as the former is the slightly better model. Overall, we see similar behavior as in fig. 4. Score always shows an overconfident island in the TP region and especially for the YOLOv3 plots, all meta regression models do not show FPs at high predicted IoU as opposed to the score.

TABLE VIII
META REGRESSION PERFORMANCE IN TERMS OF R^2 PER CONFIDENCE MODEL OVER 10-FOLD CV (std IN PARENTHESES) FOR FASTER R-CNN AND RETINANET.

	Pascal VOC	COCO	KITTI
Faster R-CNN			
Score	0.3994 (0.0002)	0.4050 (0.0001)	0.7229 (0.0002)
MC	0.2370 (0.0017)	0.2356 (0.0009)	0.4009 (0.0017)
2-Norms	0.4488 (0.0005)	0.4493 (0.0004)	0.8154 (0.0005)
1- and 2-Norms	0.5670 (0.0006)	0.4827 (0.0003)	0.8404 (0.0004)
G	0.5940 (0.0003)	0.5044 (0.0004)	0.8631 (0.0007)
MD	0.4792 (0.0009)	0.4441 (0.0004)	0.7992 (0.0004)
MD+MC	0.4920 (0.0005)	0.4454 (0.0005)	0.7999 (0.0005)
G+MC	0.6074 (0.0007)	0.5109 (0.0004)	0.8685 (0.0004)
G+MD	**0.6364 (0.0008)**	**0.5230 (0.0004)**	**0.8746 (0.0005)**
G+MD+MC	0.6302 (0.0003)	0.5221 (0.0004)	0.8699 (0.0007)
RetinaNet			
Score	0.4043 (0.0001)	0.3988 (0.0002)	0.7344 (0.0002)
MC	0.2445 (0.0009)	0.1946 (0.0006)	0.5051 (0.0012)
2-Norms	0.4619 (0.0005)	0.3801 (0.0004)	0.7917 (0.0004)
1- and 2-Norms	0.4964 (0.0006)	0.4196 (0.0005)	0.8135 (0.0005)
G	0.5723 (0.0007)	0.4774 (0.0006)	0.8447 (0.0004)
MD	0.5027 (0.0010)	0.4245 (0.0012)	0.7753 (0.0008)
MD+MC	0.5336 (0.0008)	0.4482 (0.0007)	0.7851 (0.0007)
G+MC	0.6173 (0.0004)	0.5063 (0.0007)	0.8508 (0.0004)
G+MD	**0.6432 (0.0007)**	0.5107 (0.0009)	**0.8573 (0.0009)**
G+MD+MC	0.6344 (0.0006)	**0.5138 (0.0009)**	0.8564 (0.0008)

d) MetaFusion on Pascal VOC and COCO: We show MetaFusion plots analogous to the right one of fig. 3 also for YOLOv3 on the evaluation splits of VOC and COCO in fig. 7. The qualitative behavior is analogous to the one presented for the KITTI dataset in section IV with significant improvements of G and G+MD+MC and over the score baseline in the region of low decision thresholds. Again, MetaFusion based on MC dropout performs worse than the score baseline for decision thresholds of about 0.5 and larger.

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

TABLE VII
META CLASSIFICATION PERFORMANCE IN TERMS OF $AuROC$ AND AP PER CONFIDENCE MODEL OVER 10-FOLD CV (std IN PARENTHESES) FOR FASTER-RCNN AND RETINANET.

	Pascal VOC		COCO		KITTI	
	$AuROC$	AP	$AuROC$	AP	$AuROC$	AP
Faster R-CNN						
Score	0.9331 (0.0003)	0.6771 (0.0003)	0.9128 (0.0001)	0.6414 (0.0003)	0.9792 (0.0003)	0.9329 (0.0002)
MC	0.8999 (0.0006)	0.4422 (0.0026)	0.8580 (0.0003)	0.4048 (0.0012)	0.9339 (0.0007)	0.6782 (0.0024)
$\|\cdot\|_2$	0.9104 (0.0007)	0.6166 (0.0015)	0.8980 (0.0003)	0.6116 (0.0006)	0.9875 (0.0002)	0.9301 (0.0005)
$\|\cdot\|_1, \|\cdot\|_2$	0.9491 (0.0004)	0.6773 (0.0010)	0.9064 (0.0003)	0.6253 (0.0007)	0.9897 (0.0003)	0.9389 (0.0006)
G	0.9588 (0.0005)	0.6874 (0.0013)	0.9138 (0.0003)	0.6331 (0.0007)	0.9920 (0.0001)	0.9460 (0.0007)
MD	0.9443 (0.0002)	0.7118 (0.0006)	0.9131 (0.0002)	0.6473 (0.0005)	0.9886 (0.0003)	0.9431 (0.0005)
MD+MC	0.9440 (0.0005)	0.7131 (0.0011)	0.913 (0.0002)	0.6491 (0.0003)	0.9882 (0.0003)	0.9425 (0.0005)
G+MC	0.9659 (0.0003)	0.7131 (0.0008)	0.9209 (0.0002)	0.6459 (0.0005)	0.9934 (0.0002)	0.9524 (0.0005)
G+MD	**0.9677 (0.0005)**	**0.7360 (0.0007)**	**0.9230 (0.0002)**	0.6567 (0.0005)	**0.9937 (0.0002)**	**0.9538 (0.0004)**
G+MD+MC	0.9672 (0.0004)	0.7351 (0.0010)	**0.9230 (0.0001)**	**0.6577 (0.0006)**	0.9935 (0.0002)	0.9537 (0.0003)
RetinaNet						
Score	0.8753 (0.0003)	0.6630 (0.0005)	0.8509 (0.0001)	0.6858 (0.0001)	0.9591 (0.0002)	0.8993 (0.0002)
MC	0.8576 (0.0004)	0.4181 (0.0012)	0.7696 (0.0004)	0.4354 (0.0006)	0.8813 (0.0006)	0.7119 (0.0010)
$\|\cdot\|_2$	0.8786 (0.0004)	0.6435 (0.0006)	0.8162 (0.0004)	0.6395 (0.0003)	0.9593 (0.0003)	0.9003 (0.0005)
$\|\cdot\|_1, \|\cdot\|_2$	0.8877 (0.0006)	0.6540 (0.0005)	0.8353 (0.0005)	0.6588 (0.0007)	0.9647 (0.0004)	0.9050 (0.0003)
G	0.9158 (0.0004)	0.6832 (0.0006)	0.8559 (0.0002)	0.6793 (0.0004)	0.9726 (0.0003)	0.9151 (0.0007)
MD	0.8957 (0.0004)	0.6843 (0.0007)	0.8495 (0.0004)	0.6832 (0.0006)	0.9619 (0.0003)	0.9013 (0.0004)
MD+MC	0.9006 (0.0002)	0.6906 (0.0007)	0.8520 (0.0003)	0.6846 (0.0004)	0.9627 (0.0002)	0.9022 (0.0006)
G+MC	0.9254 (0.0003)	0.7065 (0.0006)	0.8687 (0.0003)	0.6942 (0.0003)	0.9752 (0.0002)	0.9198 (0.0006)
G+MD	**0.9299 (0.0003)**	0.7230 (0.0008)	0.8715 (0.0005)	0.7016 (0.0007)	0.9761 (0.0002)	0.9226 (0.0005)
G+MD+MC	0.9295 (0.0003)	**0.7233 (0.0007)**	**0.8720 (0.0004)**	**0.7021 (0.0003)**	**0.9763 (0.0001)**	**0.9230 (0.0003)**

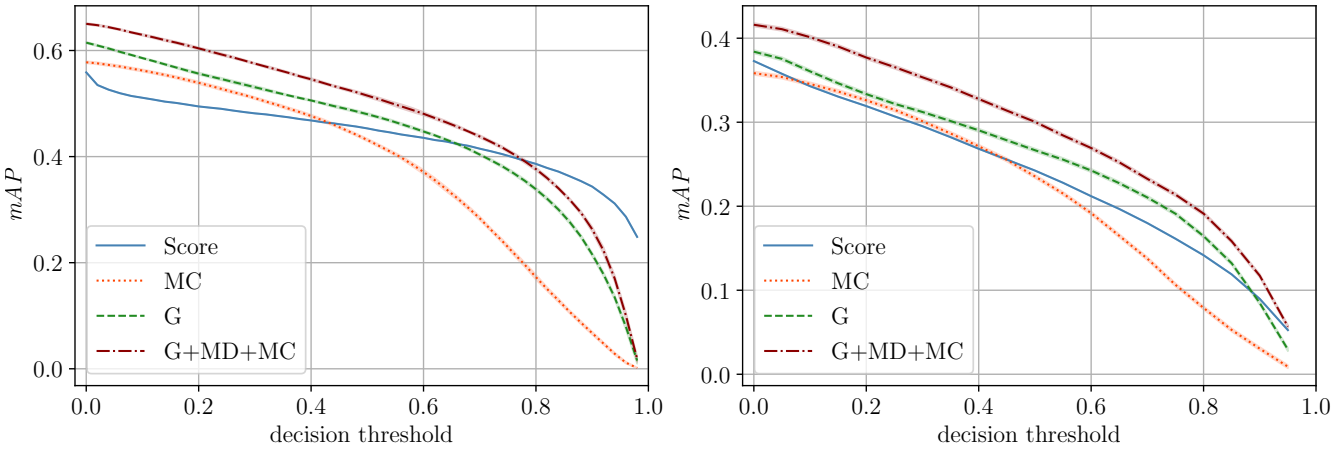

Fig. 7. Score baseline and MetaFusion mAP for the VOC (*left*) and the COCO (*right*) evaluation dataset. Model: YOLOv3.

Preprint – Preprint – Preprint – Preprint – Preprint – Preprint

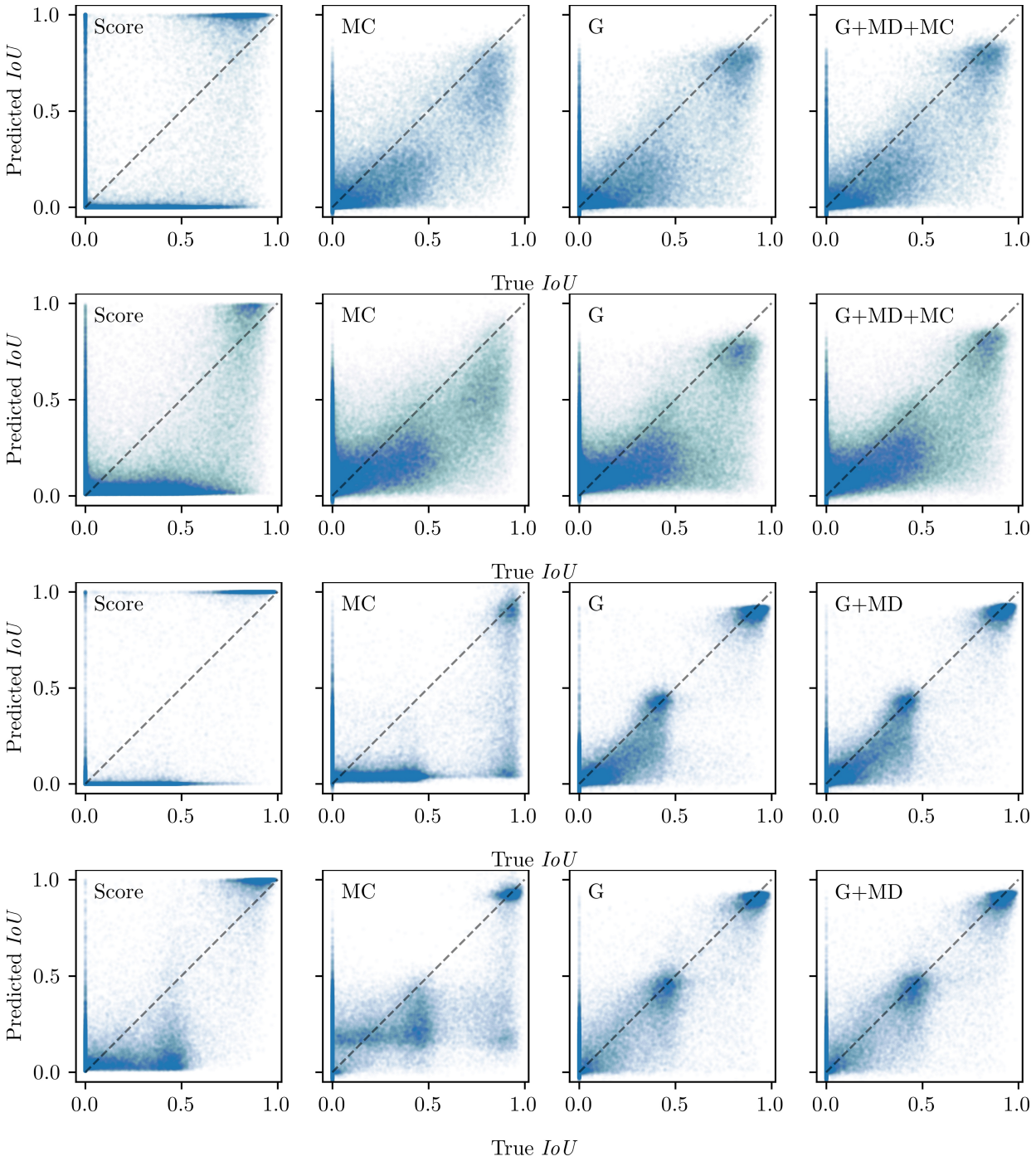

Fig. 8. Meta regression scatter plots showing predicted IoU over true IoU for four confidence models. The optimal diagonal is shown as a dashed line. From top to bottom: (YOLOv3, VOC); (YOLOv3, COCO); (Faster R-CNN, KITTI); (RetinaNet, KITTI).