Supplementary Material to

Gradient-Based Quantification of Epistemic Uncertainty for Deep Object Detectors

A. Object Detection

A.1. Notation.

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

$$\hat{y}(\boldsymbol{x}, \boldsymbol{w}) = (\hat{y}^1(\boldsymbol{x}, \boldsymbol{w}), \ldots, \hat{y}^{N_{\boldsymbol{x}}}(\boldsymbol{x}, \boldsymbol{w})) \in \mathbb{R}^{N_{\boldsymbol{x}} \times (4+1+C)} \tag{7}$$

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

$$\hat{y}^j = (\hat{\xi}^j, \hat{s}^j, \hat{p}^j) \in \mathbb{R}^{4+1+C} \tag{8}$$

consists of localizations $\hat{\xi}^j = (\hat{\mathsf{x}}^j, \hat{\mathsf{y}}^j, \hat{\mathsf{w}}^j, \hat{\mathsf{h}}^j)$ encoded e.g. as center coordinates $\hat{\mathsf{x}}$, $\hat{\mathsf{y}}$ together with width $\hat{\mathsf{w}}$ and height $\hat{\mathsf{h}}$. Moreover, a list of $N_{\boldsymbol{x}}$ integers $\hat{\kappa} \in \{1, \ldots, C\}$ 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 $\hat{p} = (\hat{p}_1, \ldots, \hat{p}_C) \in (0,1)^C$ over all C categories. Finally, $N_{\boldsymbol{x}}$ scores $\hat{s} \in (0,1)$ indicate the probability of each box being correct.

The predicted $N_{\boldsymbol{x}}$ boxes are obtained by different filtering mechanisms as a subset of a fixed number N_{out} (usually about 10^5 to 10^6) of output boxes

$$\widetilde{y}(\boldsymbol{x}, \boldsymbol{w}) = (\widetilde{y}^1, \ldots, \widetilde{y}^{N_{\text{out}}}). \tag{9}$$

The latter are the regression and classification result of predetermined "prior" or "anchor boxes". Predicted box localization $\hat{\xi}$ is usually learned as offsets and width- and height scaling of fixed anchor boxes [43, 26] or region proposals [44] (see appendix B). The most prominent examples (and the ones employed in all architectures we investigate) 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.2. 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

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{10}$$

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 $\mathrm{cand}[\hat{y}^j]$. Note, that we can also determine candidate boxes for an output box $\widetilde{y}^j$. 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.

A.3. 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 $j = 1, \ldots, N_{\boldsymbol{x}}$ 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 [7], or Adam [21] by minimizing an empirical loss

$$\mathcal{L} = \mathcal{L}_\xi + \mathcal{L}_s + \mathcal{L}_p \,. \tag{11}$$

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 ($\widetilde{\xi}$), score ($\widetilde{s}$) assignment to boxes (assigning large loss to high score for incorrect boxes and low score for correct boxes) and incorrect class probability distribution ($\widetilde{p}$), respectively. We explicitly give formulas for all utilized loss functions in appendix B. The trainable weights $\boldsymbol{w}$ of the model are updated in standard gradient descent optimization by

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} \mathcal{L}(\widetilde{y}(\boldsymbol{x}, \boldsymbol{w}), y) \tag{12}$$

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

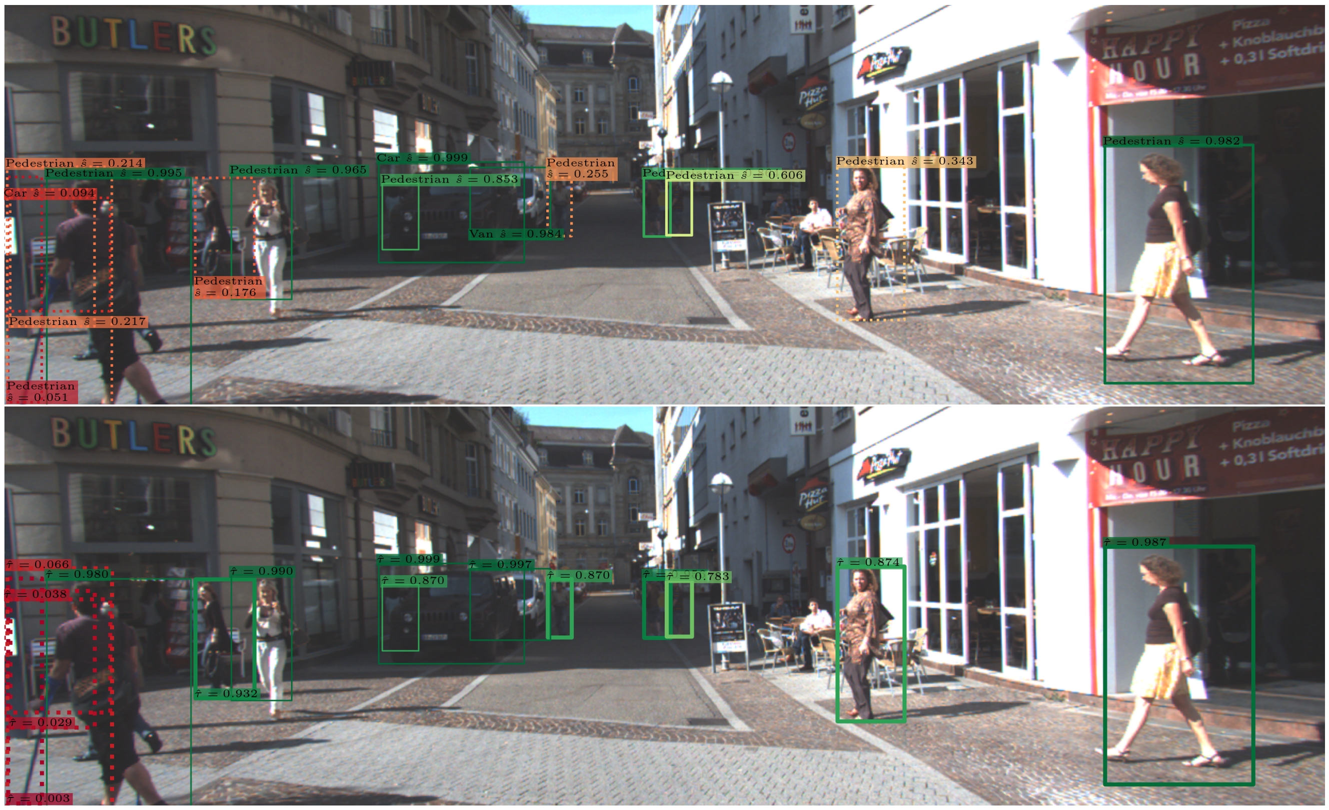

Figure 6. Full version of confidence estimation example in fig. 1. Top: DNN Score $\hat{s}$; bottom: meta classification confidence $\hat{\tau}$ involving gradient features. True predictions with low confidence are assigned large meta classification confidences while false predictions are assigned low values. This allows for improved filtering based on confidence.

A.4. Calibration.

Generally, "calibration" methods (or re-calibration) aim at rectifying scores as confidences in the sense of section 1 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 [39] where temperature scaling was found to improve calibration. In addition to considering the expected calibration error (ECE) and the maximum calibration error (MCE) [38], the authors of [39] 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 [24] introduce natural extensions to localization-dependent calibration methods and a localization-dependent metric to measure calibration for different image regions.

In Sec. 5, we evaluated the calibration of meta classifiers in terms of the maximum (MCE, [38]) and average (ACE, [39]) calibration error which we define here. We sort the examples into bins β_i, $i = 1, \ldots, B$ of a fixed width (in our case 0.1, refer to section 5) according to their confidence. For each bin β_i, we compute

$$\mathrm{acc}_i = \frac{\mathrm{TP}_i}{|\beta_i|}, \qquad \mathrm{conf}_i = \frac{1}{|\beta_i|}\sum_{j=1}^{|\beta_i|} \hat{c}_i \tag{13}$$

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. In standard classification tasks, this boils down to the classification accuracy, whereas in the object detection setting, this is the detectors precision in the bin β_i. A meta classifier performs a binary classification on detector positives, so we keep with the notation used for classifiers. Calibration metrics are usually defined as functions of the bin-wise differences between acc_i and conf_i. In particular, we investigate the following calibration error metrics:

$$MCE = \max_{i=1,\dots,B} |\mathrm{acc}_i - \mathrm{conf}_i|, \tag{14}$$

$$ACE = \frac{1}{B}\sum_{i=1}^{B} |\mathrm{acc}_i - \mathrm{conf}_i|, \tag{15}$$

$$ECE = \frac{1}{B}\sum_{i=1}^{B} \frac{B}{|\beta_i|}|\mathrm{acc}_i - \mathrm{conf}_i|. \tag{16}$$

Table 14 also shows the expected (ECE, [38]) calibration error which was argued in [39] to be biased toward bins with large amounts of examples. ECE is, thus, less informative for safety-critical investigations.

B. Implemented Loss Functions

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

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

$$\begin{aligned}\mathcal{L}_\xi^{\mathrm{Yv3}}(\hat{y}, y) =& 2\sum_{a=1}^{N_{\mathrm{out}}}\sum_{t=1}^{N_{\boldsymbol{x}}} \mathbb{I}_{at}^{\mathrm{obj}} \cdot \left[\mathrm{MSE}\left(\begin{pmatrix}\widetilde{\tau}_{\mathsf{w}}^a \\ \widetilde{\tau}_{\mathsf{h}}^a\end{pmatrix}, \begin{pmatrix}\tau_{\mathsf{w}}^t \\ \tau_{\mathsf{h}}^t\end{pmatrix}\right)\right. \\ & \left. + \mathrm{BCE}\left(\sigma\begin{pmatrix}\widetilde{\tau}_{\mathsf{x}}^a \\ \widetilde{\tau}_{\mathsf{y}}^a\end{pmatrix}, \sigma\begin{pmatrix}\tau_{\mathsf{x}}^t \\ \tau_{\mathsf{y}}^t\end{pmatrix}\right)\right],\end{aligned} \tag{17}$$

$$\begin{aligned}\mathcal{L}_s^{\mathrm{Yv3}}(\widetilde{y}, y) =& \sum_{a=1}^{N_{\mathrm{out}}}\sum_{t=1}^{N_{\boldsymbol{x}}} \Big[\mathbb{I}_{at}^{\mathrm{obj}} \mathrm{BCE}_a\left(\sigma(\widetilde{\tau}_s), \mathbf{1}_{N_{\mathrm{out}}}\right) \\ & + \mathbb{I}_{at}^{\mathrm{noobj}}(y)\mathrm{BCE}_a\left(\sigma(\widetilde{\tau}_s), \mathbf{0}_{N_{\mathrm{out}}}\right)\Big],\end{aligned} \tag{18}$$

$$\mathcal{L}_p^{\mathrm{Yv3}}(\widetilde{y}, y) = \sum_{a=1}^{N_{\mathrm{out}}}\sum_{t=1}^{N_{\boldsymbol{x}}} \mathbb{I}_{at}^{\mathrm{obj}} \mathrm{BCE}\left(\sigma(\widetilde{\tau}_p^a), \sigma(\tau_p^t)\right). \tag{19}$$

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. (22)) for the regression of the bounding box size. As introduced in [43], 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 appendix D) for one anchor denoted by

$$\widetilde{\tau} = (\widetilde{\tau}_{\mathsf{x}}, \widetilde{\tau}_{\mathsf{y}}, \widetilde{\tau}_{\mathsf{w}}, \widetilde{\tau}_{\mathsf{h}}, \widetilde{\tau}_s, \widetilde{\tau}_{p_1}, \dots, \widetilde{\tau}_{p_C}) \tag{20}$$

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

$$\begin{gathered}\widetilde{\mathsf{x}} = \ell \cdot \sigma(\widetilde{\tau}_{\mathsf{x}}) + c_{\mathsf{x}}, \quad \widetilde{\mathsf{y}} = \ell \cdot \sigma(\widetilde{\tau}_{\mathsf{y}}) + c_{\mathsf{y}}, \quad \widetilde{\mathsf{w}} = \pi_{\mathsf{w}} \cdot \mathrm{e}^{\widetilde{\tau}_{\mathsf{w}}}, \\ \widetilde{\mathsf{h}} = \pi_{\mathsf{h}} \cdot \mathrm{e}^{\widetilde{\tau}_{\mathsf{h}}}, \quad \widetilde{s} = \sigma(\widetilde{\tau}_s), \quad \widetilde{p}_j = \sigma(\widetilde{\tau}_{p_j}).\end{gathered} \tag{21}$$

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 $\widetilde{\tau}$. 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 (real) ground truth y which is collected in a feature map γ. Then, we have

$$\mathrm{MSE}(\widetilde{\tau}, \tau) = \sum_i (\widetilde{\tau}_i - \tau_i)^2. \tag{22}$$

Whenever summation indices are not clearly specified, we assume from the context that they take on all possible values, *e.g.* in eq. (17) 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:

$$\begin{aligned}\mathrm{BCE}(p, q) &= \sum_i \mathrm{BCE}_i(p, q) \\ &= -\sum_i q_i \log(p_i) + (1 - q_i)\log(1 - p_i),\end{aligned} \tag{23}$$

$$\mathrm{CE}(p, q) = -\sum_i q_i \log(p_i) \tag{24}$$

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" $\widetilde{p}_j$ are in general not normalized. Note also, that each summand in eq. (18) 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 $\mathcal{L}_\xi^{\mathrm{YOLOv3}}$.

The tensors $\mathbb{I}^{\mathrm{obj}}$ and $\mathbb{I}^{\mathrm{noobj}}$ indicate whether anchor a can be associated to ground truth instance t (obj) or not (noobj) which is determined as in [44] from two thresholds $\varepsilon_+ \geq \varepsilon_- \geq 0$ which are set to 0.5 in our implementation.

Note, that both, $\mathbb{I}^{\mathrm{obj}}$ and $\mathbb{I}^{\mathrm{noobj}}$ only depend on the ground truth y and the fixed anchors, but not on the output regression results $\widetilde{y}$ or 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 $\sigma(\tau_p^t) := p^t = (\delta_{i,\kappa_t})_{i=1}^C$.

Faster R-CNN and Cascade R-CNN. Since Faster R-CNN [44] and Cascade R-CNN [1] are two-stage architectures, 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 $\widetilde{\theta}_\xi$ for the RPN outputs $\widetilde{y}^{\mathrm{RPN}}$ and $\widetilde{\theta}_s$ the proposal score output (where

$\widetilde{s}^a = \sigma(\widetilde{\theta}^a_s)$ is the proposal score):

$$\mathcal{L}^{\mathrm{RPN}}_{\xi}(\widetilde{y}^{\mathrm{RPN}}, y) = \frac{1}{|I^+|} \sum_{a=1}^{N^{\mathrm{RPN}}_{\mathrm{out}}} \sum_{t=1}^{N_{\boldsymbol{x}}} I^+_a \widetilde{\mathbb{I}}^{\mathrm{obj}}_{at} \, \mathrm{sm}L^1_\beta \left(\widetilde{\theta}^a_\xi, \theta^t_\xi\right), \tag{25}$$

$$\begin{aligned}\mathcal{L}^{\mathrm{RPN}}_{s}(\widetilde{y}^{\mathrm{RPN}}, y) = \sum_{a=1}^{N^{\mathrm{RPN}}_{\mathrm{out}}} \sum_{t=1}^{N_{\boldsymbol{x}}} \Big[& \widetilde{\mathbb{I}}^{\mathrm{obj}}_{at} \mathrm{BCE}_a \left(\sigma(\widetilde{\theta}_s), \mathbf{1}_{N^{\mathrm{RPN}}_{\mathrm{out}}}\right) \\ & + I^-_a \widetilde{\mathbb{I}}^{\mathrm{noobj}}_{at} \mathrm{BCE}_a \left(\sigma(\widetilde{\theta}_s), \mathbf{0}_{N^{\mathrm{RPN}}_{\mathrm{out}}}\right)\Big].\end{aligned} \tag{26}$$

The tensors $\widetilde{\mathbb{I}}^{\mathrm{obj}}$ and $\widetilde{\mathbb{I}}^{\mathrm{noobj}}$ 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 $n_+ := \min\{|\widetilde{\mathbb{I}}^{\mathrm{obj}}|, b/2\}$ of the $|\widetilde{\mathbb{I}}^{\mathrm{obj}}|$ positive anchors (constituting the mask I^+) and $n_- := \min\{|\widetilde{\mathbb{I}}^{\mathrm{noobj}}|, b - n_+\}$ negative anchors (I^-). The summation of a ranges over the $N^{\mathrm{RPN}}_{\mathrm{out}}$ outputs of the RPN (in our case, 1000). Proposal regression is done based on the smooth L^1 loss

$$\mathrm{sm}L^1_\beta(\widetilde{\theta}, \theta) := \sum_i \left\{ \begin{array}{l|l} \frac{1}{2}|\widetilde{\theta}_i - \theta_i|^2 & |\widetilde{\theta}_i - \theta_i| < \beta \\ |\widetilde{\theta}_i - \theta_i| - \frac{\beta}{2} & |\widetilde{\theta}_i - \theta_i| \geq \beta \end{array} \right. , \tag{27}$$

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

$$\begin{aligned} \widetilde{\mathsf{x}} &= \pi_{\mathsf{w}} \cdot \widetilde{\tau}_{\mathsf{x}} + \pi_{\mathsf{x}}, \quad \widetilde{\mathsf{y}} = \pi_{\mathsf{y}} \cdot \widetilde{\tau}_{\mathsf{y}} + \pi_{\mathsf{y}}, \\ \widetilde{\mathsf{w}} &= \pi_{\mathsf{w}} \cdot \mathrm{e}^{\widetilde{\tau}_{\mathsf{w}}}, \quad \widetilde{\mathsf{h}} = \pi_{\mathsf{h}} \cdot \mathrm{e}^{\widetilde{\tau}_{\mathsf{h}}}, \quad \widetilde{p} = \varSigma(\widetilde{\tau}_p), \end{aligned} \tag{28}$$

where $\varSigma^i(x) := \mathrm{e}^{x_i} / \sum_j \mathrm{e}^{x_j}$ 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:

$$\mathcal{L}^{\mathrm{RoI}}_{\xi}(\widetilde{y}, y) = \frac{1}{|\mathbb{I}^{\mathrm{obj}}|} \sum_{a=1}^{N_{\mathrm{out}}} \sum_{t=1}^{N_{\boldsymbol{x}}} \mathbb{I}^{\mathrm{obj}}_{at} \, \mathrm{sm}L^1_\beta \left(\widetilde{\tau}^a_\xi, \tau^t_\xi\right), \tag{29}$$

$$\begin{aligned}\mathcal{L}^{\mathrm{RoI}}_{p}(\widetilde{y}, y) = \sum_{a=1}^{N_{\mathrm{out}}} \sum_{t=1}^{N_{\boldsymbol{x}}} \Big[& \mathbb{I}^{\mathrm{obj}}_{at} \, \mathrm{CE}(\varSigma(\widetilde{\tau}^a_p), p^t) \\ & + \mathbb{I}^{\mathrm{noobj}}_{at} \, \mathrm{CE}(\varSigma(\widetilde{\tau}^a_{p_0}), 1)\Big].\end{aligned} \tag{30}$$

[1]The result $\widetilde{\tau}$ is similar to eq. (20), but without $\widetilde{\tau}_s$ where we have instead $C+1$ classes, with one "background" class. We denote the respective probability by $\widetilde{p}_0$.

Here, $\mathbb{I}^{\mathrm{obj}}$ and $\mathbb{I}^{\mathrm{noobj}}$ are computed with $\varepsilon_+ = \varepsilon_- = 0.5$. The cascaded bounding box regression of Cascade R-CNN implements the smooth L^1 loss at each of three cascade stages, where bounding box offsets and scaling are computed from the previous bounding box regression results as proposals.

RetinaNet. In the RetinaNet [26] architecture, score assignment is part of the classification.

$$\mathcal{L}^{\mathrm{Ret}}_{\xi}(\widetilde{y}, y) = \frac{1}{|\mathbb{I}^{\mathrm{obj}}|} \sum_{a=1}^{N_{\mathrm{out}}} \sum_{t=1}^{N_{\boldsymbol{x}}} \mathbb{I}^{\mathrm{obj}}_{at} L^1 \left(\widetilde{\tau}^a_\xi, \tau^t_\xi\right), \tag{31}$$

$$\begin{aligned}\mathcal{L}^{\mathrm{Ret}}_{p}(\widetilde{y}, y) = \frac{1}{|\mathbb{I}^{\mathrm{obj}}|} \sum_{a=1}^{N_{\mathrm{out}}} \sum_{t=1}^{N_{\boldsymbol{x}}} \Big[& \mathbb{I}^{\mathrm{obj}}_{at} \sum_{j=1}^{C} \alpha(1 - \sigma(\widetilde{\tau}^a_{p_j}))^{\gamma_{\mathrm{F}}} \cdot \\ & \cdot \mathrm{BCE}_j \left(\sigma(\widetilde{\tau}^a_p), p^t\right) \\ & + \mathbb{I}^{\mathrm{noobj}}_{at} (1-\alpha) \sigma(\widetilde{\tau}^a_{p_0})^{\gamma_{\mathrm{F}}} \cdot \mathrm{BCE}\left(\sigma(\widetilde{\tau}^a_{p_0}), 0\right)\Big].\end{aligned} \tag{32}$$

For $\mathbb{I}^{\mathrm{obj}}$ and $\mathbb{I}^{\mathrm{noobj}}$, we use $\varepsilon_+ = 0.5$ and $\varepsilon_- = 0.4$. Regression is based on the absolute loss $L^1(\widetilde{\tau}, \tau) = \sum_i |\widetilde{\tau}_i - \tau_i|$ and the classification loss is a formulation of the well-known focal loss with $\alpha = 0.25$ and $\gamma_{\mathrm{F}} = 2$. Bounding box transformation follows the maps in eq. (28), where π are the respective RetinaNet anchor localizations instead of region proposals. The class-wise scores of the prediction are obtained by $\widetilde{p}_j = \sigma(\widetilde{\tau}_{p_j})$, $j = 1, \ldots, C$ as for YOLOv3.

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 appendix D. Note, that for all derivatives of the cross entropy, we can use

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\left[-y \log(\sigma(\tau)) - (1-y)\log(1-\sigma(\tau))\right] = \sigma(\tau) - y. \tag{33}$$

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

$$\frac{\partial}{\partial \widetilde{\tau}^b_r} \mathcal{L}^{\mathrm{Yv3}}_{\xi} = 2 \sum_{t=1}^{N_{\boldsymbol{x}}} \mathbb{I}^{\mathrm{obj}}_{bt} \left\{ \begin{array}{l|l} \widetilde{\tau}^b_r - \tau^t_r & r \in \{\mathsf{w}, \mathsf{h}\} \\ \sigma(\widetilde{\tau}^b_r) - \sigma(\tau^t_r) & r \in \{\mathsf{x}, \mathsf{y}\} \\ 0 & \text{otherwise.} \end{array} \right. , \tag{34}$$

$$\frac{\partial}{\partial \widetilde{\tau}^b_r} \mathcal{L}^{\mathrm{Yv3}}_{s} = \delta_{rs} \sum_{t=1}^{N_{\boldsymbol{x}}} \left[\mathbb{I}^{\mathrm{obj}}_{bt}(\widetilde{s}^b - 1) + \mathbb{I}^{\mathrm{noobj}}_{bt} \widetilde{s}^b\right], \tag{35}$$

$$\frac{\partial}{\partial \widetilde{\tau}^b_r} \mathcal{L}^{\mathrm{Yv3}}_{p} = \sum_{t=1}^{N_{\boldsymbol{x}}} \mathbb{I}^{\mathrm{obj}}_{bt} \sum_{i=1}^{C} \delta_{r p_i} (\widetilde{p}^b_i - p^t_i), \tag{36}$$

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

$$\frac{\partial}{\partial \hat{\theta}_r^b} \mathcal{L}_\xi^{\mathrm{RPN}} = \frac{1}{|I^+|} \sum_{t=1}^{N_{\boldsymbol{x}}} I_b^+ \tilde{\mathbb{I}}_{bt}^{\mathrm{obj}} \cdot \begin{cases} \hat{\theta}_r^b - \theta_r^t & \begin{array}{|l} |\hat{\theta}_r^b - \theta_r^t| < \beta \text{ and} \\ \quad r \in \{\mathsf{x}, \mathsf{y}, \mathsf{w}, \mathsf{h}\} \end{array} \\ \mathrm{sgn}(\hat{\theta}_r^b - \theta_r^t) & \begin{array}{|l} |\hat{\theta}_r^b - \theta_r^t| \geq \beta \text{ and} \\ \quad r \in \{\mathsf{x}, \mathsf{y}, \mathsf{w}, \mathsf{h}\} \end{array} \\ 0 & \begin{array}{|l} \text{otherwise} \end{array} \end{cases}, \tag{37}$$

$$\frac{\partial}{\partial \hat{\theta}_r^b} \mathcal{L}_s^{\mathrm{RPN}} = \delta_{rs} \left[I_b^+ \tilde{\mathbb{I}}_{bt}^{\mathrm{obj}} (\hat{s}^b - 1) + I_b^- \tilde{\mathbb{I}}_{bt}^{\mathrm{noobj}} \hat{s}^b \right]. \tag{38}$$

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

$$\frac{\partial}{\partial \hat{\tau}_r^b} \mathcal{L}_\xi^{\mathrm{RoI}} = \frac{1}{|\mathbb{I}^{\mathrm{obj}}|} \sum_{t=1}^{N_{\boldsymbol{x}}} \mathbb{I}_{bt}^{\mathrm{obj}} \cdot \begin{cases} \hat{\tau}_r^b - \tau_r^t & \begin{array}{|l} |\hat{\tau}_r^b - \tau_r^t| < \beta \text{ and} \\ \quad r \in \{\mathsf{x}, \mathsf{y}, \mathsf{w}, \mathsf{h}\} \end{array} \\ \mathrm{sgn}(\hat{\tau}_r^b - \tau_r^t) & \begin{array}{|l} |\hat{\tau}_r^b - \tau_r^t| \geq \beta \text{ and} \\ \quad r \in \{\mathsf{x}, \mathsf{y}, \mathsf{w}, \mathsf{h}\} \end{array} \\ 0 & \begin{array}{|l} \text{otherwise} \end{array} \end{cases}, \tag{39}$$

$$\begin{aligned} \frac{\partial}{\partial \hat{\tau}_r^b} \mathcal{L}_p^{\mathrm{RoI}} = - \sum_{t=1}^{N_{\boldsymbol{x}}} \Bigg[& \mathbb{I}_{bt}^{\mathrm{obj}} \sum_{j=1}^{C} p_j^t \left(\delta_{p_j r} - \sum_{k=0}^{C} \delta_{p_k r} \varSigma^k (\hat{\tau}_p^b) \right) \\ & + \mathbb{I}_{bt}^{\mathrm{noobj}} \left(\delta_{p_0 r} - \sum_{k=0}^{C} \delta_{p_k r} \varSigma^k (\hat{\tau}_p^b) \right) \Bigg]. \end{aligned} \tag{40}$$

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

$$\frac{\partial}{\partial \tilde{\tau}_r^b} \mathcal{L}_\xi^{\mathrm{Ret}} = \frac{1}{|\mathbb{I}^{\mathrm{obj}}|} \sum_{t=1}^{N_{\boldsymbol{x}}} \mathbb{I}_{bt}^{\mathrm{obj}} \cdot \begin{cases} \mathrm{sgn}(\tilde{\tau}_r^b - \tau_r^t) & \begin{array}{|l} r \in \{\mathsf{x}, \mathsf{y}, \mathsf{w}, \mathsf{h}\} \end{array} \\ 0 & \begin{array}{|l} \text{otherwise} \end{array} \end{cases}, \tag{41}$$

$$\begin{aligned} \frac{\partial}{\partial \tilde{\tau}_r^b} \mathcal{L}_p^{\mathrm{Ret}} = \frac{1}{|\mathbb{I}^{\mathrm{obj}}|} \sum_{t=1}^{N_{\boldsymbol{x}}} \Bigg[& \mathbb{I}_{bt}^{\mathrm{obj}} \sum_{j=1}^{C} \delta_{p_j r} \alpha (1 - \sigma(\tilde{\tau}_{p_j}^b))^{\gamma_{\mathrm{F}}} \cdot \\ & \cdot [-\gamma_{\mathrm{F}} \sigma(\tilde{\tau}_{p_j}^b) \mathrm{BCE}_j(\sigma(\tilde{\tau}_p^b), p^t) + \sigma(\tilde{\tau}_{p_j}^b) - 1] \\ & + \mathbb{I}_{bt}^{\mathrm{noobj}} \delta_{p_0 r} (1 - \alpha) \sigma(\tilde{\tau}_{p_0}^b)^{\gamma_{\mathrm{F}}} \cdot \\ & \cdot [-\gamma_{\mathrm{F}} (1 - \sigma(\tilde{\tau}_{p_0}^b)) \log(1 - \sigma(\tilde{\tau}_{p_0}^b)) + \sigma(\tilde{\tau}_{p_0}^b)] \Bigg]. \end{aligned} \tag{42}$$

Table 6. Dataset splits used for training and evaluation of object detectors. Note, that we train meta classifiers and meta regressors on a validation part of the evaluation split and evaluate it on the complementary split.

Dataset	training	evaluation	# eval images
VOC	2007+2012 trainval	2007 test	4952
COCO	train2017	val2017	5000
KITTI	random part of training	complement part of training	2000

C. Implementation details

Here, we state details of the implementations of our framework to different architectures, and on different datasets.

C.1. Datasets

In order to show a wide range of applications, we investigate our method on the following object detection datasets (see table 6 for the splits used). Meta classification and meta regression models are as post-processing modules fitted on a validation sample of the evaluation dataset and their performance is evaluated on the complementary sample of the evaluation dataset (in cross-validation).

Pascal VOC 2007+2012 [8]. The Pascal VOC dataset is an object detection benchmark of everyday images involving 20 different object categories. We train on the 2007 and 2012 trainval splits, accumulating to 16550 train images and we evaluate on the 2007 test split of 4952 images. For training, we include labels marked as "difficult" in the original annotations.

MS COCO 2017 [27]. The MS COCO dataset constitutes a second vision benchmark involving 2D bounding box detection annotations for everyday images with 80 object categories. We train on the train2017 split of 118287 images and evaluate on the 5000 images of the val2017 split.

KITTI [11]. The KITTI vision benchmark contains 21 real world street scenes annotated with 2D bounding boxes. We randomly divide the 7481 labeled images into a training split of 5481 images and use the complement of 2000 images for evaluation.

C.2. Detectors

For our experiments, we employ three common object detection architectures, namely YOLOv3 with Darknet53 backbone [43], Faster R-CNN [44] and RetinaNet [26], each with a ResNet50FPN [15] backbone. Moreover, we investigate a state-of-the-art detector in Cascade R-CNN [1] with a large ResNeSt200FPN [57] backbone. We started from PyTorch [41] reimplementations, added dropout layers and trained from scratch on the datasets in table 6. We list some of the detector-specific details.

YOLOv3. The basis of our implementation is a publicly available GitHub repository [55]. We position dropout layers with $p = 0.5$ before the last convolutional layers of each detection head. Gradient features are computed over the last two layers in each of the three detection heads as the final network layers have been found to be most informative in the classification setting [40]. Since each output box is the result of exactly one of the three heads, we only have two layers for gradients per box resulting in 2×3 gradients per box (2 layers per 3 losses) as indicated in table 1. We train an ensemble of 5 detectors for each dataset from scratch.

Faster R-CNN. Based on the official Torchvision implementation, our model uses dropout ($p = 0.5$) before the last fully connected layer of the architecture (classification and bounding box prediction in the Fast R-CNN head). We compute gradient features for the last two fully connected layers of the Fast R-CNN head as well as for the last two convolutional layers of the RPN per box (objectness and localization), leading to 4×2 gradients per box ($2 + 2$ for localization, 2 for classification and 2 for proposal objectness).

RetinaNet. We also employ RetinaNet as implemented in Torchvision with ($p = 0.5$)-dropout before the last convolutional layers for bounding box regression and classification. Gradients are computed for the last two convolutional layers for bounding box regression and classification resulting in 2×2 gradients per prediction.

Cascade R-CNN. We use the Detectron2[56]-supported implementation of ResNeSt provided by the ResNeSt authors Zhang *et al*. [57] and the pre-trained weights on the MS COCO dataset. We train from scratch on Pascal VOC and KITTI. Since this model is primarily interesting for investigation due to its naturally strong score baseline based on cascaded regression, we do not report MC dropout results for it. Gradient uncertainty features are computed for the last two fully connected layers (bounding box regression and classification) of each of the three cascades. The loss of later cascade stages depends in principle on the weights of previous cascade stages. However, we only compute the gradients with respect to the weights in the current stage resulting in 2×6 (3 stages for bounding box regression and classification) gradients for the Cascade R-CNN head. Furthermore, we have the 2×2 RPN gradients as in Faster R-CNN.

C.3. Uncertainty baselines

We give a short account of the baselines implemented and investigated in our experiments.

Score. By the score, we mean the box-wise objectness score for YOLOv3 and the maximum softmax probability for Faster R-CNN, RetinaNet and Cascade R-CNN. As standard object detection pipelines discard output bounding boxes based on a score threshold, this quantity is the baseline for discriminating true against false outputs.

Entropy. The entropy is a common "hand-crafted" uncertainty measure based on the classification output $\widetilde{p} \in [0, 1]^C$ (softmax or category-wise sigmoid) and given by

$$H(\widetilde{p}) = -\sum_{c=1}^{C} \widetilde{p}_c \log(\widetilde{p}_c). \tag{43}$$

Energy. As an alternative to the maximum softmax probability and the entropy, Liu *et al*. proposed an energy score depending on a temperature parameter T given by

$$E(\widetilde{\tau}) = -T \log \sum_{c=1}^{C} \mathrm{e}^{\widetilde{\tau}_{p_c}/T} \tag{44}$$

based on the probability logits $(\widetilde{\tau}_{p_1}, \ldots, \widetilde{\tau}_{p_C})$. We found that $T = 100$ delivers the strongest results, see table 7 where we compared different values of T (like in [29]) for YOLOv3 on the KITTI dataset in terms of meta classification and meta regression performance.

Table 7. Ablation on the temperature parameter T for the energy score in terms of meta classification ($AuROC$ and AP) and meta regression (R^2).

	$AuROC$	AP	R^2
$T = 1$	92.52 ± 0.03	91.86 ± 0.04	62.12 ± 0.09
$T = 10$	78.42 ± 0.13	81.75 ± 0.08	32.92 ± 0.20
$T = 100$	95.66 ± 0.02	95.33 ± 0.03	71.79 ± 0.06
$T = 1000$	95.62 ± 0.03	95.33 ± 0.04	71.78 ± 0.05
Score	96.53 ± 0.05	96.87 ± 0.03	78.86 ± 0.05
MD	$\mathbf{98.23 \pm 0.02}$	$\mathbf{98.06 \pm 0.02}$	$\mathbf{85.88 \pm 0.10}$
GS_{full}	$\underline{98.04 \pm 0.03}$	$\underline{97.81 \pm 0.06}$	$\underline{85.40 \pm 0.11}$

Table 8. Ablation on the sample count size N_{MC} for MC dropout in terms of meta classification ($AuROC$ and AP) and meta regression (R^2). Results obtained from the sample standard deviation.

	$AuROC$	AP	R^2
$N_{\text{MC}} = 10$	97.40 ± 0.04	96.91 ± 0.06	80.85 ± 0.10
$N_{\text{MC}} = 15$	97.50 ± 0.03	97.08 ± 0.07	81.28 ± 0.09
$N_{\text{MC}} = 20$	97.69 ± 0.03	97.28 ± 0.05	82.11 ± 0.09
$N_{\text{MC}} = 25$	97.64 ± 0.03	97.20 ± 0.04	81.94 ± 0.12
$N_{\text{MC}} = 30$	97.60 ± 0.07	97.17 ± 0.10	82.10 ± 0.11
$N_{\text{MC}} = 35$	97.71 ± 0.03	97.29 ± 0.05	82.17 ± 0.13
$N_{\text{MC}} = 40$	97.69 ± 0.04	97.29 ± 0.06	82.12 ± 0.13
Score	96.53 ± 0.05	96.87 ± 0.03	78.86 ± 0.05
MD	$\mathbf{98.23 \pm 0.02}$	$\mathbf{98.06 \pm 0.02}$	$\mathbf{85.88 \pm 0.10}$
GS_{full}	$\underline{98.04 \pm 0.03}$	$\underline{97.81 \pm 0.06}$	$\underline{85.40 \pm 0.11}$

Full softmax. We investigate an enveloping model of all classification-based uncertainty features by involving all probabilities $(\widetilde{p}_1, \ldots, \widetilde{p}_C)$ directly as co-variables in the meta classifier or meta regression model. We find that it outperforms all purely classification-based models, which is expected.

MC dropout (MC). As a common baseline, we investigate Monte-Carlo dropout uncertainty. Since we are explicitly interested in the uncertainty content of MC dropout, we only include anchor-wise standard deviations of the entire

Table 9. Ablation on the ensemble size N_{ens} for deep ensembles in terms of meta classification ($AuROC$ and AP) and meta regression (R^2). Results obtained from the sample standard deviation.

	$AuROC$	AP	R^2
$N_{\text{ens}} = 3$	97.53 ± 0.03	97.17 ± 0.05	82.63 ± 0.13
$N_{\text{ens}} = 4$	97.79 ± 0.04	97.48 ± 0.06	83.62 ± 0.12
$N_{\text{ens}} = 5$	97.92 ± 0.04	97.63 ± 0.05	84.18 ± 0.12
$N_{\text{ens}} = 6$	98.04 ± 0.03	97.75 ± 0.04	84.64 ± 0.16
$N_{\text{ens}} = 7$	98.06 ± 0.03	97.80 ± 0.05	84.78 ± 0.11
$N_{\text{ens}} = 8$	$\underline{98.08 \pm 0.02}$	97.80 ± 0.03	84.91 ± 0.10
Score	96.53 ± 0.05	96.87 ± 0.03	78.86 ± 0.05
MD	$\mathbf{98.23 \pm 0.02}$	$\mathbf{98.06 \pm 0.02}$	$\mathbf{85.88 \pm 0.10}$
GS_{full}	98.04 ± 0.03	$\underline{97.81 \pm 0.06}$	$\underline{85.40 \pm 0.11}$

network output $\widetilde{y}$ obtained from 30 dropout samples. We found that computing more samples does not significantly improve predictive uncertainty content as seen in the ablation study on the MC sample count N_{MC} in table 8 for YOLOv3 on the KITTI dataset. Meta classification performance can be further improved by involving dropout means of $\widetilde{y}$. However, MC dropout means do not carry an intrinsic meaning of uncertainty as opposed to standard deviations, so we do not include them in our main experiments.

Deep ensembles (E). As another common, sampling-based baseline, we investigate deep ensemble uncertainty obtained from ensembles of size 5. We find that larger ensembles do not significantly improve meta classification performance. For reference, we show an ablation on the ensemble size N_{ens} for YOLOv3 on the KITTI dataset in table 9 in terms of meta classification and meta regression. By the same motivation like for MC dropout, we only include anchor-wise standard deviations over forward passes from the ensemble.

MetaDetect (MD). The output-based MetaDetect framework computes uncertainty features for use in meta classification and meta regression from pre-NMS variance in anchor-based object detection. In our implementation, we compute the $46 + C$ (where C is the number of categories) MetaDetect features[48] which include the entire network output $\widetilde{y}$. The MetaDetect framework is, therefore, an enveloping model to any uncertainty features based on the object detection output (in particular to any classification-based uncertainty) which we also find in our experiments. We include it in order to cover all such baselines.

Details of gradient-based uncertainty (GS)**.** In our experiments, we investigate two gradient-based uncertainty models. While $\text{GS}_{\|\cdot\|_2}$ is based on the two-norms of box-wise gradients, GS_{full} is utilizes all the six maps in eq. (3). While the two norms $\|\cdot\|_1$ and $\|\cdot\|_2$ directly compute the magnitude of a vector, the maps $\text{mean}(\cdot)$ and $\text{std}(\cdot)$ do not immediately capture a concept of length. However, they have been found in [40] to yield decent separation capabilities. Similarly, the component-wise $\min(\cdot)$ and $\max(\cdot)$ contain relevant predictive information. Note, that the latter two are

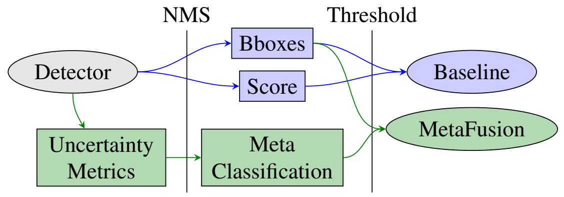

Figure 7. Schematic sketch of the baseline detection pipeline and the alternative MetaFusion pipeline for an object detector.

related to the sup-norm $\|\cdot\|_\infty$ but together contain more information. While the last layer gradients themselves are highly informative, we allow for gradients of the last two layers in our main experiments. In table 10 we show meta classification and meta regression performance of gradient-based models with features obtained from different numbers of network layers of the YOLOv3 model on the KITTI dataset. Starting with the last layer gradient only (# layers is 1), the gradient features from the two last layers and so on. We see that meta classification performance quickly saturates and no significant benefit can be seen from using more than 3 layers. However, meta regression can still be improved slightly by using up to 5 network layers.

In some of our experiments, we compute gradients via the PyTorch autograd framework, iteratively for each bounding box. While this alleviates significant implementation effort, this procedure is computationally far less efficient than directly computing the gradients from the formulas in appendix B as is done in our runtime measurements.

In order to save on computational effort, we compute gradient features 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 ~ 150 predictions per image. These settings lead to a highly disbalanced TP/FP ratio post NMS on which meta classification and meta regression models are fitted. On YOLOv3, for example, these ratios are for Pascal VOC: 0.099, MS COCO: 0.158 and for KITTI: 0.464, so our models fit on significantly more FPs than TPs. However, our meta classification and meta regression models (see section 4) are gradient boosting models which tend to reflect well-calibrated confidences / regressions on the domain of training data. Our results (*e.g.* table 2) obtained from cross-validation confirm that this ratio does not constitute an obstacle for obtaining well-performing models on data not used to fit the model. For gradient boosting models, we employ the XGBoost library [3] with 30 estimators (otherwise standard settings).

C.4. MetaFusion framework

In section 5, we showed a way of trading uncertainty information for detection performance. Figure 7 shows a sketch of the resulting pipeline, where the usual object de-

Table 10. Ablation on the number of network layers used in terms of meta classification ($AuROC$ and AP) and meta regression (R^2). Gradient features per layer are accumulated to those of later layers starting from the last layer of the DNN.

Metric	Score	# layers 1	2	3	4	5
$AuROC$	96.53 ± 0.05	98.04 ± 0.03	98.06 ± 0.02	98.18 ± 0.03	98.18 ± 0.03	98.19 ± 0.02
AP	96.87 ± 0.03	97.81 ± 0.06	97.83 ± 0.04	97.98 ± 0.05	98.00 ± 0.04	98.04 ± 0.04
R^2	78.89 ± 0.05	84.35 ± 0.05	85.40 ± 0.11	86.04 ± 0.11	86.18 ± 0.07	86.24 ± 0.09

tection pipeline is shown in blue. The standard object detection pipeline relies on filtering out false positive output boxes on the basis of their score (see also fig. 6). An altered confidence estimation like meta classification can improve the threshold-dependent detection quality of the object detection pipeline. This way, boxes which are falsely assigned a low score can survive the thresholding step. Similarly, FPs with a high score may be suppressed by proper predictive confidence estimation methods. This approach is not limited to meta classification, however, our experiments show that meta classification constitutes such a method.

D. 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 features are usually computed via backpropagation only for a few network layers. Therefore, we restrict ourselves to the setting of fully convolutional neural networks. **Setting.** As in [49, 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

$$(K_{t+1})_c^d \in \mathbb{R}^{(2s_t+1)\times(2s_t+1)}, \quad c \in [k_t]; \quad d \in [k_{t+1}], \tag{45}$$

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

$$(K * \phi)_{ab} := \sum_{m,n=-s}^{s} K_{s+1+p,s+1+q}\phi_{a+p,b+q}, \tag{46}$$

where $a = 1, \dots, h$ and $b = 1, \dots, w$. 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

$$\psi_{t+1}^d = \sum_{c=1}^{k_t} (K_{t+1})_c^d * \phi_t^c + b_{t+1}^d \mathbf{1}_{h_t \times w_t}, \quad d \in [k_{t+1}]. \tag{47}$$

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

$$\begin{aligned} &C^{K_t} : \mathbb{R}^{k_{t-1} \times h_{t-1} \times w_{t-1}} \to \mathbb{R}^{k_t \times h_t \times w_t}, \\ &(C^{K_t}\phi_{t-1})_{ab}^d := \left(\sum_{c=1}^{k_t} (K_t)_c^d * \phi_t^c \right)_{ab}, \end{aligned} \tag{48}$$

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[2] 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 sub-net starting at layer t by ℓ_t, *i.e.*

$$\ell_T(\phi_T) := \mathcal{L}(\phi_T, \gamma), \qquad \ell_{t-1}(\phi_{t-1}) := \ell_t(\alpha_t(\psi_t)). \tag{49}$$

Straight-forward calculations yield

$$\begin{aligned} \nabla_{K_T}\mathcal{L} &= \nabla_{K_T}\left(\ell_T \circ \alpha_T \circ \psi_T(K_T)\right) \\ &= D_1\mathcal{L}|_{\phi_T} \cdot D\alpha_T|_{\psi_T} \cdot \nabla_{K_T}\psi_T \end{aligned} \tag{50}$$

$$\begin{aligned} \nabla_{K_{T-1}}\mathcal{L} &= \nabla_{K_{T-1}}\left(\ell_T \circ \alpha_T \circ \psi_T \circ \alpha_{T-1} \circ \psi_{T-1}(K_{T-1})\right) \\ &= D_1\mathcal{L}|_{\phi_T} \cdot D\alpha_T|_{\psi_T} \cdot C^{K_T} \\ &\quad \cdot D\alpha_{T-1}|_{\psi_{T-1}} \cdot \nabla_{K_{T-1}}\psi_{T-1}. \end{aligned} \tag{51}$$

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 4, we omitted the terms $D\alpha_T|_{\psi_T}$ and

[2]The transformations are listed in appendix B for the entries τ of ϕ_T.

$D\alpha_{T-1}|_{\psi_{T-1}}$. We will come back to them later in the discussion. For the gradient features we present in this paper, each $\widetilde{y}^j$ 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 $\widetilde{y}^j$ (see appendix 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 $\overline{y}^j$ in feature map representation. We address next, how this masking influences eq. (50), eq. (51) and the FLOP count of our method.

Computing the mask. The complexity of determining μ^j (*i.e.* finding $\mathrm{cand}[\widetilde{y}^j]$) is the complexity of computing all mutual IoU values between $\widetilde{y}^j$ 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:

$$U = A_1 + A_2 - I, \qquad IoU = I/U, \tag{52}$$

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[3] μ^j in eq. (50) and eq. (51) 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 $\widetilde{y}^j$.

In table 11 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 appendix B. In principle, those formulas allow for every possible choice of $b \in [N_{\text{out}}]$ 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 $\widetilde{y}^j$ in a separate step involving an IoU computation between $\widetilde{y}^j$ and the respective anchor. The total count of candidate boxes in practice is on average not larger than ~ 30. When evaluating the formulas from appendix 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 (42). In table 11 we also list the additional post-processing cost for the output transformations (see appendix B, eqs. (21) and (28)) required for sampling-based uncertainty quantification like MC dropout or deep ensemble samples ("sampling pp"). The latter are also proportional to N_{out}, but also to the number N_{samp} of samples.

[3]See section 4. The mask μ^j selects the feature map representation of $\mathrm{cand}[\widetilde{y}^j]$ out of ϕ_T.

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

Theorem 2 *The number of FLOP required to compute the last layer ($t = T$) 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 t,* 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 variational inference only on the last layer,* i.e. *ϕ_{T-1} requires $\mathcal{O}(k_T k_{T-1} hw)$ FLOP per sample.*

Our implementations exclusively use stride 1 convolutions for the layers indicated in section 5, 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. (50). 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. (47)), so $\nabla_{K_t}\psi_t$ 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 $c \in [k_t]$, $d \in [k_{t-1}]$, $p, q \in \{-s_t, \ldots, s_t\}$, $a \in [h_t]$ and $b \in [w_t]$. One easily sees from eqs. (46) and (47) that

$$\frac{\partial}{\partial((K_t)_c^d)_{pq}} (\psi_t)_{ab}^d = (\phi_{t-1})_{a+p-s_t-1, b+q-s_t-1}^c, \tag{53}$$

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

$$1 < a \leq 2s_t + 2, \qquad 1 < b \leq 2s_t + 2 \tag{54}$$

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. (50) 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

$$[2 \cdot k_T(2s_T + 1)^2 - 1] \cdot [k_{T-1} \cdot (2s_T + 1)^2] \tag{55}$$

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. (51), 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

Table 11. Upper bounds on FLOP and elementary function evaluations performed during the computation of $D\mathcal{L}^j$ (all contributions) and post processing for sampling-based uncertainty quantification (sampling pp) for N_{samp} inference samples.

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

$\phi \in \mathbb{R}^{n_{t-1}}$ from the left via

$$\begin{aligned}\left(C^{K_t}\phi\right)_{ab}^{d} &= \sum_{c=1}^{k_{t-1}} \left[(K_t)_c^d * \phi^c\right]_{ab} \\ &= \sum_{c=1}^{k_{t-1}} \sum_{m,n=-s_t}^{s_t} [(K_t)_c^d]_{s_t+1+m, s_t+1+n} (\phi^c)_{a+m,b+n},\end{aligned} \tag{56}$$

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

$$\begin{aligned}\nabla_{K_{t+1}}\mathcal{L} =& \nabla_{K_{t+1}}\ell_{t+1}(\phi_{t+1}(K_{t+1})) \\ =& D\ell_{t+1}|_{\phi_{t+1}} \cdot \alpha_{t+1}|_{\psi_{t+1}} \cdot \nabla_{K_{t+1}}\psi_{t+1},\end{aligned} \tag{57}$$

then by backpropagation, *i.e.* eq. (49), we obtain

$$\begin{aligned}\nabla_{K_t}\mathcal{L} &= \nabla_{K_t}[\ell_{t+1} \circ \alpha_{t+1} \circ \psi_{t+1}(\phi_t(K_t))] \\ &= D\ell_{t+1}|_{\phi_{t+1}} \cdot \alpha_{t+1}|_{\psi_{t+1}} \cdot C^{K_{t+1}} \cdot D\alpha_t|_{\psi_t} \cdot \nabla_{K_t}\psi_t.\end{aligned} \tag{58}$$

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

$$\begin{aligned}&[2 \cdot k_{t+1}(2s_{t+1}+1)^2 - 1] \cdot [k_t(2s_t+1)^2] \\ &+ [2 \cdot k_t(2s_t+1)^2 - 1] \cdot [k_{t-1}(2s_t+1)^2]\end{aligned} \tag{59}$$

via the backpropagation step from $\nabla_{K_{t+1}}\mathcal{L}$. The claim in theorem 1 addressing eq. (51), follows for $t = T-1$ in eq. (59).

Finally, we address the computational complexity for sampling-based uncertainty quantification methods with sampling on ϕ_{T-1}. This is applicable, *e.g.*, for dropout on the last layer (as in our experiments) or a deep sub-ensemble [53] sharing the forward pass up to the last layer (note, that we do not use sub-ensembles in our experiments, but regular deep ensembles). Earlier sampling leads to far higher FLOP counts. Again, we ignore the cost of dropout itself as it is random binary masking together with a respective upscaling/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

$$2n_T k_{T-1}(2s_T+1)^2 - 1 + n_T \tag{60}$$

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

Discussion. A large part of the FLOP required to compute gradient features results from the computation of the masks μ^j and the term $D\mathcal{L}^j$ for each relevant predicted box. In table 11 we have treated the latter separately and found that, although the counts listed for $D\mathcal{L}^j$ apply to each separate box, sampling post-processing comes with considerable computational complexity as well. In that regard, we have similar costs for gradient features and sampling over the last network layer. Note in particular, that computing $D\mathcal{L}^j$ requires no new evaluation of elementary functions, as opposed to sampling. Once $D\mathcal{L}^j$ is computed for $\widetilde{y}^j$, 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 sample results in $\mathcal{O}(n_T k_{T-1})$ with sampling on ϕ_{T-1}. Sampling any earlier results in additional full convolution forward passes which also come with considerable computational costs. We note that sampling-based epistemic uncertainty can be computed in parallel with all N_{samp} forward passes being performed simultaneously. Gradient uncertainty features, in contrast, require one full forward pass for the individual gradients $\nabla_{K_t}\mathcal{L}(\mu^j\phi_T(K_t), \gamma^j)$ to be computed. Therefore, gradient uncertainty features experience a slight computational latency as compared to sampling methods. We argue that in principle, all following steps (computation of μ^j and $\nabla_{K_t}\mathcal{L}(\mu^j\phi_T(K_t), \gamma^j)$) 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 sampling-based methods, 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. (50) and (51) for a leaky ReLU activation only leads to lower-order terms. The same terms were also omitted before in determining the computational complexity of sampling uncertainty methods. Also, for this consideration, we regard the fully connected layers used for bounding box regression and classification in the Faster/Cascade R-CNN RoI head as (1×1)-convolutions to stay in the setting presented here.

E. Theoretical Link with Empirical Findings.

Here, we elaborate on the points raised in section 3 by considering data $(y, x) \sim p(y, x) = p(y|x)p(x)$ and a parametric classification model

$$f_w : x \mapsto f(y|x, w) = \hat{y}(x, w) \tag{61}$$

estimating $p(y|x)$. The labels y and the model's class prediction $\overline{y} = \mathrm{argmax}_{c \in \mathcal{C}} f(y|x, w)$ are categorical over a class space $\mathcal{C} = \{1, \ldots, C\}$. For a loss function $\mathcal{L} = \mathcal{L}((y, x), f(\cdot|\cdot, w))$ (*e.g.*, the negative log-likelihood $\mathcal{L}((y, x), f(\cdot|\cdot, w)) = -\log(f(y|x, w))$), we compute the gradient

$$g(x) = g(x, w, \overline{y}) = \nabla_w \mathcal{L}((\overline{y}, x), f(\cdot|\cdot, w)) \tag{62}$$

where we neglect the implicit w-dependency in $\overline{y}(x)$ since $\overline{y}(x)$ is locally constant with discontinuity on decision boundaries of $f(\cdot|\cdot, w)$. This self-learning gradient coincides with the ordinary learning gradient $g(x, w, y)$ with frequency $P(y = \overline{y}(x))$ (the accuracy of the model).

We can investigate whether $g(x)$ is large (in the sense of some metric M like $M(g) = \|g\|_\rho$ for $\rho \in [1, \infty]$) statistically whenever the prediction $\overline{y}(x)$ is uncertain / prone to error. Thus, we compare conditional distributions over $M(g(x))$ conditioned to incorrect predictions $\overline{y}(x) \neq y$ versus correct predictions $\overline{y} = y$. The application of different risk functionals to the distributions can then relate the statistical magnitudes of $M(g)$ conditional to $\overline{y}$ (larger functional values for statistically larger values of $M(g)$). Investigating the expected value as a simple risk functional, we search conditions for

$$\mathbb{E}_{(y,x)}[M(g(x))|\overline{y}(x) = y] < \mathbb{E}_{(y,x)}[M(g(x))|\overline{y} \neq y]. \tag{63}$$

We first call $\varepsilon(x)$ the conditional error rate and ε the total error rate of the model f_w under the distribution $p(y, x)$ with

$$\varepsilon(x) = \sum_{c \neq \overline{y}(x)} p(c|x), \quad \varepsilon = \mathbb{E}_{x \sim p(x)}[\varepsilon(x)]. \tag{64}$$

The conditional expectations then yield

$$\begin{aligned}
&\mathbb{E}_{(y,x)\sim p(y,x)}[M(g(x))|\overline{y}(x) \neq y] \\
&\quad = \frac{\int \sum_{y \neq \overline{y}(x)} p(y|x) M(g(x))\, \mathrm{d}p(x)}{P(\overline{y}(x) \neq y)} \\
&\quad = \frac{1}{\varepsilon} \mathbb{E}_{x \sim p(x)}[\varepsilon(x) M(g(x))] \\
&\quad = \mathbb{E}[M(g(x))] + \frac{\mathrm{Cov}(\varepsilon(x), M(g(x)))}{\varepsilon}
\end{aligned} \tag{65}$$

$$\begin{aligned}
&\mathbb{E}_{(y,x)\sim p(y,x)}[M(g(x))|\overline{y}(x) = y] \\
&\quad = \frac{\int (1 - \varepsilon(x)) M(g(x))\, \mathrm{d}p(x)}{P(\overline{y}(x) = y)} \\
&\quad = \frac{1}{1 - \varepsilon} \mathbb{E}_{x \sim p(x)}[(1 - \varepsilon(x)) M(x)] \\
&\quad = \mathbb{E}[M(g(x))] + \frac{\mathrm{Cov}(\varepsilon(x), M(g(x)))}{\varepsilon - 1}.
\end{aligned} \tag{66}$$

Therefore, $\mathbb{E}[M(g(x))]$ and the total error rate ε drop out of eq. (63), which is finally equivalent to

$$\mathrm{Cov}(\varepsilon(x), M(g(x))) > 0. \tag{67}$$

For accurate models f, the self-learning gradient $g(x)$ will be close to the real gradient $g(x, w, y)$, so the above covariance will be close to $\mathrm{Cov}(\varepsilon(x), M(g(x, w, y)))$. The positivity of the latter has a clear interpretation in terms of epistemic uncertainty, in that learning steps are larger whenever the model performance is poor ($\varepsilon(x)$ is large). In such regions, the model still attempts to adapt strongly to new instances, whereas little adaptation is given for instances where the model performance is already good ($\varepsilon(x)$ is small). That such local improvement are in fact possible is an easy consequence of the universal approximation property of deep neural networks.

That the condition for the self-learning gradient in (67) holds can be seen in the classification experiments conducted and shown in [40, Fig. 2] and [51, Fig. 2] where the distributions of $M(g(x))$ are explicitly conditioned to true and false predictions.

F. Further Numerical Results

F.1. Non-redundancy with output-based uncertainty

Gradient features show significant improvements when combined with output- or sampling-based uncertainty quantification methods (see table 2 and table 3). We show additional meta classification and meta regression results in table 12 and in table 13 to further illustrate this finding. First, in table 12 we find that adding $\mathrm{GS}_{\mathrm{full}}$ to the raw object detection output features $\widetilde{y}$ performs similarly as the combination $\mathrm{GS}_{\mathrm{full}}$+MD. In fact, when directly comparing MD

Table 12. Meta classification ($AuROC$ and AP) and meta regression (R^2) performance of baseline methods, variants of gradient metrics and different combinations of output-based uncertainty quantification methods with gradient metrics ($\text{mean} \pm \text{std}$). We also show the results of using the entire network output $\widetilde{y}$ for meta classification and regression, as well, as adding sampling means to standard deviation features for MC and E.

	Pascal VOC			COCO			KITTI		
YOLOv3	$AuROC$	AP	R^2	$AuROC$	AP	R^2	$AuROC$	AP	R^2
Score	90.68 ± 0.06	69.56 ± 0.12	48.29 ± 0.04	82.97 ± 0.04	62.31 ± 0.05	32.60 ± 0.02	96.55 ± 0.04	96.87 ± 0.03	78.83 ± 0.05
Entropy	91.30 ± 0.02	61.94 ± 0.06	43.24 ± 0.03	76.52 ± 0.02	42.52 ± 0.04	21.10 ± 0.04	94.78 ± 0.03	94.82 ± 0.05	69.33 ± 0.08
Energy	92.59 ± 0.02	64.65 ± 0.06	47.18 ± 0.03	75.39 ± 0.02	39.72 ± 0.06	17.94 ± 0.02	95.46 ± 0.05	94.63 ± 0.08	70.39 ± 0.10
Full Softmax	93.81 ± 0.06	72.08 ± 0.15	53.86 ± 0.11	82.91 ± 0.06	58.65 ± 0.10	36.95 ± 0.13	97.10 ± 0.02	96.90 ± 0.04	78.79 ± 0.12
Full output $\widetilde{y}$	95.84 ± 0.04	78.84 ± 0.10	60.67 ± 0.18	86.31 ± 0.05	67.46 ± 0.07	44.32 ± 0.11	98.35 ± 0.02	98.21 ± 0.04	86.34 ± 0.07
MC$_{\text{std}}$	96.72 ± 0.02	78.15 ± 0.09	61.63 ± 0.15	89.04 ± 0.02	64.94 ± 0.11	43.85 ± 0.09	95.43 ± 0.04	94.11 ± 0.12	75.09 ± 0.13
MC$_{\text{std+mean}}$	97.42 ± 0.02	84.18 ± 0.09	68.33 ± 0.16	90.40 ± 0.03	72.63 ± 0.07	52.38 ± 0.07	98.43 ± 0.03	98.28 ± 0.04	86.86 ± 0.09
E$_{\text{std}}$	96.87 ± 0.02	77.86 ± 0.11	61.48 ± 0.07	88.97 ± 0.02	64.05 ± 0.12	43.53 ± 0.13	97.98 ± 0.03	97.69 ± 0.04	84.29 ± 0.12
E$_{\text{std+mean}}$	97.62 ± 0.02	84.87 ± 0.14	68.88 ± 0.09	90.75 ± 0.03	73.15 ± 0.06	53.09 ± 0.09	98.61 ± 0.02	$\underline{98.49 \pm 0.03}$	88.00 ± 0.08
MC$_{\text{std+mean}}$+E$_{\text{std+mean}}$	97.69 ± 0.02	85.30 ± 0.11	69.60 ± 0.13	91.15 ± 0.03	73.85 ± 0.05	54.12 ± 0.09	98.61 ± 0.01	$\underline{98.49 \pm 0.02}$	87.95 ± 0.10
MD	95.78 ± 0.05	78.64 ± 0.08	60.36 ± 0.14	86.23 ± 0.05	67.37 ± 0.08	44.22 ± 0.11	98.23 ± 0.03	98.07 ± 0.03	85.97 ± 0.09
GS$_{\|\cdot\|_2}$	94.76 ± 0.03	74.86 ± 0.10	58.05 ± 0.13	84.90 ± 0.02	61.49 ± 0.08	38.77 ± 0.04	97.30 ± 0.05	96.82 ± 0.10	81.11 ± 0.14
GS$_{\|\cdot\|_{1,2}}$	95.03 ± 0.03	76.04 ± 0.10	59.83 ± 0.10	86.21 ± 0.04	63.32 ± 0.13	41.36 ± 0.09	97.65 ± 0.04	97.21 ± 0.07	83.27 ± 0.09
GS$_{\text{full}}$	95.80 ± 0.04	78.57 ± 0.11	62.50 ± 0.11	86.94 ± 0.04	66.96 ± 0.06	44.90 ± 0.09	98.04 ± 0.02	97.81 ± 0.04	85.28 ± 0.07
GS$_{\text{full}}$+$\widetilde{y}$	96.51 ± 0.018	81.20 ± 0.09	65.24 ± 0.16	87.54 ± 0.04	69.05 ± 0.07	47.67 ± 0.09	98.57 ± 0.03	98.47 ± 0.04	87.83 ± 0.08
GS$_{\text{full}}$+MC$_{\text{std}}$	97.65 ± 0.01	85.12 ± 0.06	70.30 ± 0.08	90.76 ± 0.02	72.50 ± 0.08	52.71 ± 0.07	98.35 ± 0.04	98.16 ± 0.04	86.48 ± 0.11
GS$_{\text{full}}$+E$_{\text{std}}$	$\underline{97.85 \pm 0.02}$	$\underline{85.90 \pm 0.15}$	$\underline{71.22 \pm 0.07}$	$\underline{91.27 \pm 0.03}$	73.44 ± 0.06	$\underline{54.17 \pm 0.06}$	$\underline{98.64 \pm 0.02}$	$\underline{98.49 \pm 0.03}$	$\underline{88.34 \pm 0.10}$
GS$_{\text{full}}$+MD	96.46 ± 0.04	81.00 ± 0.16	65.08 ± 0.14	87.51 ± 0.02	68.98 ± 0.08	47.63 ± 0.10	98.53 ± 0.03	98.42 ± 0.04	87.69 ± 0.06
MC$_{\text{std}}$+E$_{\text{std}}$+MD	97.66 ± 0.02	85.13 ± 0.12	69.38 ± 0.11	91.14 ± 0.02	$\underline{73.82 \pm 0.05}$	54.07 ± 0.08	98.56 ± 0.03	98.45 ± 0.03	87.78 ± 0.11
GS$_{\text{full}}$+MC$_{\text{std}}$+E$_{\text{std}}$+MD	$\mathbf{97.95 \pm 0.02}$	$\mathbf{86.69 \pm 0.09}$	$\mathbf{72.26 \pm 0.08}$	$\mathbf{91.65 \pm 0.03}$	$\mathbf{74.88 \pm 0.07}$	$\mathbf{56.14 \pm 0.11}$	$\mathbf{98.74 \pm 0.02}$	$\mathbf{98.62 \pm 0.01}$	$\mathbf{88.80 \pm 0.07}$

Table 13. Meta classification ($AuROC$ and AP) and meta regression (R^2) performance of baseline methods, variants of gradient metrics and combinations of output- and gradient-based metrics for different object detection architectures ($\text{mean} \pm \text{std}$).

	Pascal VOC			COCO			KITTI		
	$AuROC$	AP	R^2	$AuROC$	AP	R^2	$AuROC$	AP	R^2
Faster R-CNN									
Score	89.77 ± 0.05	67.71 ± 0.03	39.94 ± 0.02	83.82 ± 0.03	64.14 ± 0.03	40.50 ± 0.01	96.53 ± 0.05	93.29 ± 0.02	72.29 ± 0.02
MC	89.99 ± 0.06	44.22 ± 0.26	23.70 ± 0.17	85.80 ± 0.03	40.48 ± 0.12	23.56 ± 0.09	93.39 ± 0.07	67.82 ± 0.24	40.09 ± 0.17
MD	94.43 ± 0.02	71.18 ± 0.06	47.92 ± 0.09	91.31 ± 0.02	64.73 ± 0.05	44.41 ± 0.04	98.86 ± 0.03	94.31 ± 0.05	79.92 ± 0.04
GS$_{\|\cdot\|_2}$	91.04 ± 0.07	61.66 ± 0.15	44.88 ± 0.05	89.80 ± 0.03	61.16 ± 0.06	44.93 ± 0.04	98.75 ± 0.02	93.01 ± 0.05	81.54 ± 0.05
GS$_{\|\cdot\|_{1,2}}$	94.91 ± 0.04	67.73 ± 0.10	56.70 ± 0.06	90.64 ± 0.03	62.53 ± 0.07	48.27 ± 0.03	98.97 ± 0.03	93.89 ± 0.07	84.04 ± 0.04
GS$_{\text{full}}$	95.88 ± 0.05	68.74 ± 0.13	59.40 ± 0.03	91.38 ± 0.03	63.31 ± 0.07	50.44 ± 0.04	99.20 ± 0.01	94.60 ± 0.07	86.31 ± 0.07
GS$_{\text{full}}$+MC	96.59 ± 0.03	71.31 ± 0.08	60.74 ± 0.07	92.09 ± 0.02	64.59 ± 0.06	51.09 ± 0.04	99.34 ± 0.02	95.24 ± 0.05	86.85 ± 0.04
GS$_{\text{full}}$+MD	$\underline{96.77 \pm 0.05}$	$\underline{73.60 \pm 0.07}$	$\underline{63.64 \pm 0.08}$	$\underline{92.30 \pm 0.02}$	$\underline{65.67 \pm 0.05}$	$\underline{52.30 \pm 0.04}$	$\underline{99.37 \pm 0.02}$	$\underline{95.38 \pm 0.05}$	$\underline{87.46 \pm 0.05}$
GS$_{\text{full}}$+MC+MD	$\mathbf{96.72 \pm 0.04}$	$\mathbf{73.51 \pm 0.10}$	$\mathbf{63.02 \pm 0.03}$	$\mathbf{92.30 \pm 0.01}$	$\mathbf{65.77 \pm 0.06}$	$\mathbf{52.21 \pm 0.04}$	$\mathbf{99.35 \pm 0.02}$	$\mathbf{95.37 \pm 0.03}$	$\mathbf{86.99 \pm 0.07}$
RetinaNet									
Score	87.53 ± 0.03	66.30 ± 0.05	40.43 ± 0.01	84.95 ± 0.04	68.58 ± 0.01	39.88 ± 0.02	95.91 ± 0.02	89.93 ± 0.02	73.44 ± 0.02
MC	72.90 ± 0.08	27.39 ± 0.11	14.17 ± 0.12	76.96 ± 0.04	43.54 ± 0.06	19.46 ± 0.06	88.13 ± 0.06	71.19 ± 0.10	50.51 ± 0.12
MD	89.57 ± 0.04	68.43 ± 0.08	50.27 ± 0.10	85.09 ± 0.01	68.32 ± 0.06	42.45 ± 0.12	96.19 ± 0.03	90.13 ± 0.04	77.53 ± 0.08
GS$_{\|\cdot\|_2}$	87.86 ± 0.04	64.35 ± 0.06	46.19 ± 0.05	81.62 ± 0.04	63.95 ± 0.03	38.01 ± 0.04	95.93 ± 0.03	90.03 ± 0.05	79.17 ± 0.04
GS$_{\|\cdot\|_{1,2}}$	88.77 ± 0.06	65.40 ± 0.05	49.64 ± 0.06	83.53 ± 0.05	65.88 ± 0.07	41.96 ± 0.05	96.47 ± 0.04	90.50 ± 0.03	81.35 ± 0.05
GS$_{\text{full}}$	91.58 ± 0.04	68.32 ± 0.06	57.23 ± 0.07	85.59 ± 0.02	67.93 ± 0.04	47.74 ± 0.06	97.26 ± 0.03	91.51 ± 0.07	84.47 ± 0.04
GS$_{\text{full}}$+MC	92.54 ± 0.03	70.65 ± 0.06	61.73 ± 0.04	86.87 ± 0.03	69.42 ± 0.03	50.63 ± 0.07	97.52 ± 0.02	91.98 ± 0.06	85.08 ± 0.04
GS$_{\text{full}}$+MD	$\mathbf{92.99 \pm 0.03}$	$\underline{72.30 \pm 0.08}$	$\mathbf{64.32 \pm 0.07}$	$\underline{87.15 \pm 0.05}$	$\underline{70.16 \pm 0.07}$	$\underline{51.07 \pm 0.09}$	$\underline{97.61 \pm 0.02}$	$\underline{92.26 \pm 0.05}$	$\mathbf{85.73 \pm 0.09}$
GS$_{\text{full}}$+MC+MD	$\underline{92.95 \pm 0.03}$	$\mathbf{72.33 \pm 0.07}$	$\underline{63.44 \pm 0.06}$	$\mathbf{87.20 \pm 0.04}$	$\mathbf{70.21 \pm 0.03}$	$\mathbf{51.38 \pm 0.09}$	$\mathbf{97.63 \pm 0.01}$	$\mathbf{92.30 \pm 0.03}$	$\underline{85.64 \pm 0.08}$
Cascade R-CNN									
Score	95.70 ± 0.04	79.62 ± 0.10	57.90 ± 0.09	94.11 ± 0.01	81.36 ± 0.02	56.32 ± 0.02	98.67 ± 0.02	95.81 ± 0.04	83.31 ± 0.03
MD	96.32 ± 0.05	$\underline{82.11 \pm 0.12}$	63.62 ± 0.12	$\underline{94.12 \pm 0.03}$	$\underline{81.60 \pm 0.05}$	$\underline{58.84 \pm 0.04}$	99.18 ± 0.01	$\underline{96.60 \pm 0.05}$	86.22 ± 0.08
GS$_{\|\cdot\|_2}$	96.46 ± 0.05	76.94 ± 0.19	61.56 ± 0.12	93.30 ± 0.02	76.40 ± 0.06	54.13 ± 0.06	99.19 ± 0.01	95.83 ± 0.06	85.80 ± 0.06
GS$_{\|\cdot\|_{1,2}}$	96.54 ± 0.06	78.19 ± 0.22	62.82 ± 0.15	93.63 ± 0.02	77.95 ± 0.06	56.24 ± 0.05	99.23 ± 0.01	96.07 ± 0.05	86.33 ± 0.06
GS$_{\text{full}}$	$\underline{96.66 \pm 0.05}$	78.97 ± 0.19	$\underline{63.94 \pm 0.13}$	93.97 ± 0.02	79.17 ± 0.09	57.86 ± 0.05	$\underline{99.34 \pm 0.01}$	96.48 ± 0.04	$\underline{87.39 \pm 0.08}$
GS$_{\text{full}}$+MD	$\mathbf{97.24 \pm 0.05}$	$\mathbf{84.11 \pm 0.13}$	$\mathbf{69.78 \pm 0.13}$	$\mathbf{94.78 \pm 0.02}$	$\mathbf{82.53 \pm 0.05}$	$\mathbf{62.13 \pm 0.05}$	$\mathbf{99.48 \pm 0.01}$	$\mathbf{97.27 \pm 0.04}$	$\mathbf{89.59 \pm 0.04}$

with $\widetilde{y}$, we see consistently better results on $\widetilde{y}$, even though MD contains $\widetilde{y}$ as co-variables. We attribute this finding to overfitting of the gradient boosting classifier and regression on MD. This suggests that the information in MD is mostly redundant with the network output features. Also, for combinations of one output-based uncertainty source (*i.e.* one of MC, E and MD) we gain strong boosts, especially in meta regression (R^2). Note, that GS$_{\text{full}}$+E$_{\text{std}}$ is

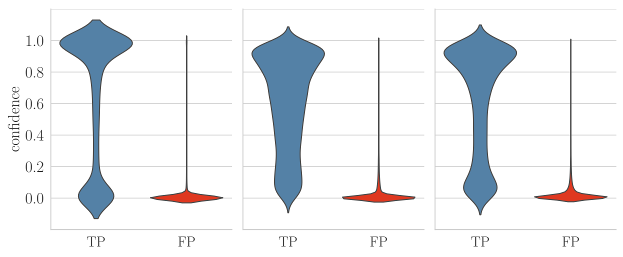

Figure 8. Confidence violin plots divided into TP and FP for Score (left), $\mathrm{GS_{full}}$ (center) and $\mathrm{GS_{full}}$+MC+E+MD (right). Model: YOLOv3, dataset: Pascal VOC evaluation split.

almost always the second-best model, even out-performing the purely output-based model $\mathrm{MC_{std}}$+$\mathrm{E_{std}}$+MD. We show meta classification and meta regression performance of the sampling-based epistemic uncertainty methods MC and E when we include sampling averages of all features in addition to standard deviations which also leads to significant boosts. Finally, we show an additional subset of $\mathrm{GS_{full}}$ consisting of one- and two-norms ($\{\|\cdot\|_1, \|\cdot\|_2\}$) of all gradients which we abbreviate by $\mathrm{GS}_{\|\cdot\|_{1,2}}$. We notice significant gain of the latter to $\mathrm{GS}_{\|\cdot\|_2}$, which shows that the one-norms $\|\cdot\|_1$ contains important predictive information. Moreover, $\mathrm{GS_{full}}$ is still significantly stronger than $\mathrm{GS}_{\|\cdot\|_{1,2}}$, showing that the other uncertainty features in eq. (3) lead to large performance boosts. Note that in almost all cases, combining MC dropout and deep ensemble features shows improvement over the single models even though both are epistemic (model) uncertainty. The two methods, therefore, do not contain the exact same information but still complement each other to some degree and are rather different approximations of epistemic uncertainty

For further illustration of our method, table 13 shows additional meta classification and meta regression results for the architectures from table 4. We find similar tendencies for the purely norm-based gradient model $\mathrm{GS}_{\|\cdot\|_{1,2}}$ and see a significant degree of non-redundancy between gradient-based uncertainty and output-based uncertainty quantification methods. Note in particular, that MC stays roughly on par with the score baseline in terms of $AuROC$. We see significantly worse performance in terms of AP and meta regression (R^2). We attribute this to the anchor-based dropout sampling method which was also employed for the present architectures (in the case of Faster R-CNN, the aggregation approach is proposal-based).

Figure 8 shows the confidence violin plots of the score (left), $\mathrm{GS_{full}}$ (center) and $\mathrm{GS_{full}}$+MC+E+MD (right) conditioned on TP and FP predictions. The violin widths are normalized for increased width contrast. The score TP-violin shows especially large density at low confidences whereas the TP-violins of $\mathrm{GS_{full}}$ and $\mathrm{GS_{full}}$+MC+E+MD are less concentrated around the confidence $\hat{\tau} = 0$. Instead, they have mass shifted towards the medium confidence range ("neck").

F.2. Calibration of meta classifiers

For sake of completeness, we list in table 14 the calibration metrics ECE, MCE and ACE defined in appendix A for score and meta classifiers for all object detectors on all three datasets investigated in section 5. All calibration metrics are in line with the results from section 5 with meta classifiers being always better calibrated than the score by at least half an order of magnitude in any calibration metric. See also fig. 9 for the additional reliability diagrams for MC, E and MC+E+MD$\mathrm{GS_{full}}$ which extends fig. 3. 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 [39], the former is also less informative as bin-wise accuracy is weighted with the bin counts. In table 14 we can see a weakly increasing trend of calibration errors in the meta classifiers due to overfitting on the increasing number of co-variables. All meta classifiers are well-calibrated across the board.

F.3. Meta regression scatter plots

We underline the meta regression results obtained in section 5 and appendix F by showing samples of predicted IoU values over their true IoU in fig. 10. The samples are the results of one cross-validation split from table 3 and we indicate the diagonal of optimal regression with a dashed line in each panel. Note that the x-axis shows the true IoU values and we indicate the uncertainty quantification method below each panel plot at a label. The y-axis shows the predicted IoU for each method. We find a large cluster for the score with low score but medium to high true IoU (from 0.1 to 0.8), the right-most part of which (predicted $IoU \geq 0.5$) are false negative predictions. In this regard, we refer again to fig. 6 where FNs such as these become very apparent. Moreover, the score indicates very little correlation with the true IoU for true $IoU \geq 0.6$ where there are numerous samples with a score between 0.4 and 0.6.

In contrast, the meta regression models show striking amounts of FPs (true IoU equal to 0 and, *e.g.*, prediction $\iota \geq 0.3$). This phenomenon seems especially apparent for Monte Carlo dropout uncertainty. The meta regression models MD, $\mathrm{GS_{full}}$ and $\mathrm{GS_{full}}$+MC+E+MD show fits that are comparatively close to the optimal diagonal which is in line with the determined regression performance R^2 between 0.81 and 0.89 in table 3.

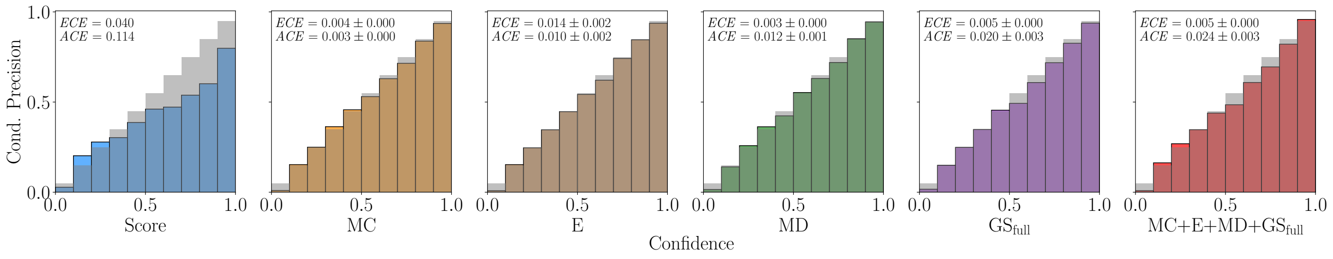

Figure 9. Reliability diagrams for the Score and meta classifiers based on different epistemic uncertainty features of the YOLOv3 architecture on the KITTI dataset. See table 14 for calibration errors of all meta classification models investigated in section 5.

Table 14. Expected (ECE, [38]), maximum (MCE, [38]) and average (ACE, [39]) calibration errors per confidence model over 10-fold cv (mean ± std).

	Pascal VOC			COCO			KITTI		
YOLOv3	ECE	MCE	ACE	ECE	MCE	ACE	ECE	MCE	ACE
Score	0.040	0.252	0.114	0.0327	0.050	0.034	0.068	0.348	0.227
Entropy	0.002 ± 0.001	0.021 ± 0.010	0.007 ± 0.003	**0.002 ± 0.001**	0.028 ± 0.020	0.007 ± 0.003	**0.005 ± 0.001**	**0.033 ± 0.010**	**0.011 ± 0.003**
Energy Score	**0.001 ± 0.001**	**0.015 ± 0.007**	**0.005 ± 0.002**	**0.002 ± 0.001**	0.021 ± 0.003	0.008 ± 0.001	0.006 ± 0.002	0.034 ± 0.010	0.013 ± 0.005
Full Softmax	0.003 ± 0.000	0.028 ± 0.006	0.010 ± 0.002	0.003 ± 0.001	0.018 ± 0.003	0.007 ± 0.001	0.008 ± 0.001	0.048 ± 0.010	0.018 ± 0.002
MC	0.004 ± 0.000	0.033 ± 0.006	0.014 ± 0.002	0.004 ± 0.001	0.025 ± 0.003	0.010 ± 0.001	0.011 ± 0.001	0.036 ± 0.010	0.013 ± 0.002
E	0.003 ± 0.000	0.025 ± 0.005	0.010 ± 0.002	0.004 ± 0.001	0.022 ± 0.003	0.010 ± 0.001	0.013 ± 0.001	0.062 ± 0.010	0.028 ± 0.004
MD	0.003 ± 0.000	0.040 ± 0.009	0.012 ± 0.001	0.005 ± 0.001	0.033 ± 0.005	0.014 ± 0.001	0.012 ± 0.001	0.074 ± 0.020	0.028 ± 0.005
$GS_{\|\cdot\|_2}$	0.003 ± 0.000	0.036 ± 0.007	0.014 ± 0.001	**0.002 ± 0.000**	**0.013 ± 0.004**	**0.005 ± 0.001**	0.008 ± 0.001	0.054 ± 0.010	0.022 ± 0.003
GS_{full}	0.005 ± 0.000	0.055 ± 0.002	0.021 ± 0.003	0.005 ± 0.001	0.039 ± 0.003	0.015 ± 0.001	0.012 ± 0.001	0.078 ± 0.020	0.034 ± 0.006
MC+E+MD	0.005 ± 0.001	0.049 ± 0.010	0.020 ± 0.003	0.005 ± 0.000	0.031 ± 0.006	0.014 ± 0.001	0.014 ± 0.001	0.076 ± 0.010	0.034 ± 0.005
MC+E+MD+GS_{full}	0.005 ± 0.000	0.061 ± 0.010	0.024 ± 0.003	0.006 ± 0.000	0.042 ± 0.004	0.018 ± 0.001	0.015 ± 0.001	0.106 ± 0.020	0.043 ± 0.006
Faster R-CNN									
Score	0.050	0.427	0.232	0.075	0.212	0.138	0.036	0.283	0.114
MD	**0.003 ± 0.000**	0.039 ± 0.007	0.013 ± 0.002	**0.004 ± 0.000**	**0.020 ± 0.003**	**0.009 ± 0.001**	**0.009 ± 0.001**	**0.079 ± 0.020**	**0.029 ± 0.004**
GS_{full}	0.004 ± 0.000	**0.027 ± 0.007**	**0.011 ± 0.001**	**0.004 ± 0.001**	0.024 ± 0.003	**0.009 ± 0.001**	0.010 ± 0.001	0.084 ± 0.020	0.035 ± 0.004
MD+GS_{full}	0.005 ± 0.000	0.044 ± 0.007	0.018 ± 0.002	0.006 ± 0.001	0.029 ± 0.006	0.012 ± 0.001	0.011 ± 0.001	0.088 ± 0.010	0.037 ± 0.004
RetinaNet									
Score	0.068	0.212	0.123	0.089	0.192	0.106	0.027	0.097	0.043
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**
GS_{full}	**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 ± 0.010	0.022 ± 0.004
MD+GS_{full}	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 ± 0.010	0.028 ± 0.003
Cascade R-CNN									
Score	0.020	0.219	0.090	0.029	0.082	0.042	0.013	0.188	0.078
MD	**0.003 ± 0.000**	**0.021 ± 0.006**	**0.007 ± 0.002**	**0.003 ± 0.000**	0.019 ± 0.007	**0.006 ± 0.001**	**0.002 ± 0.000**	**0.038 ± 0.010**	**0.016 ± 0.005**
GS_{full}	0.005 ± 0.000	0.032 ± 0.010	0.012 ± 0.002	**0.003 ± 0.000**	**0.017 ± 0.003**	0.007 ± 0.001	0.003 ± 0.000	0.052 ± 0.010	0.020 ± 0.004
MD+GS_{full}	0.005 ± 0.000	0.034 ± 0.008	0.014 ± 0.002	0.004 ± 0.000	0.025 ± 0.004	0.010 ± 0.001	0.003 ± 0.000	0.046 ± 0.009	0.019 ± 0.003

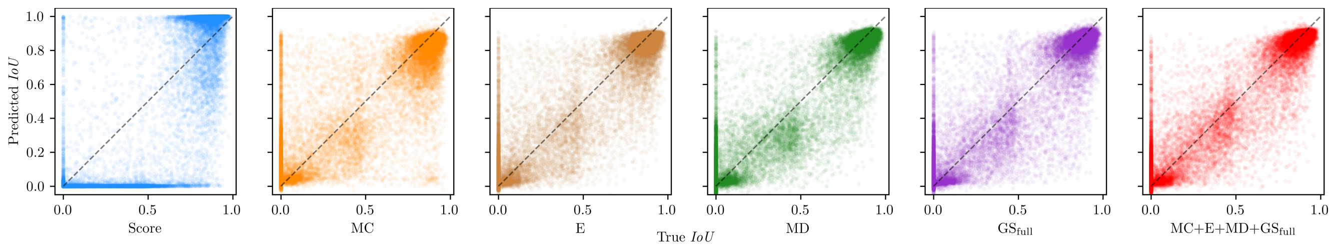

Figure 10. Scatter plots for samples of Score and meta regression based on MC dropout, gradient features G and the combination model G+MD+MC. We draw the optimal diagonal for reference. Model: YOLOv3, dataset: KITTI evaluation split.