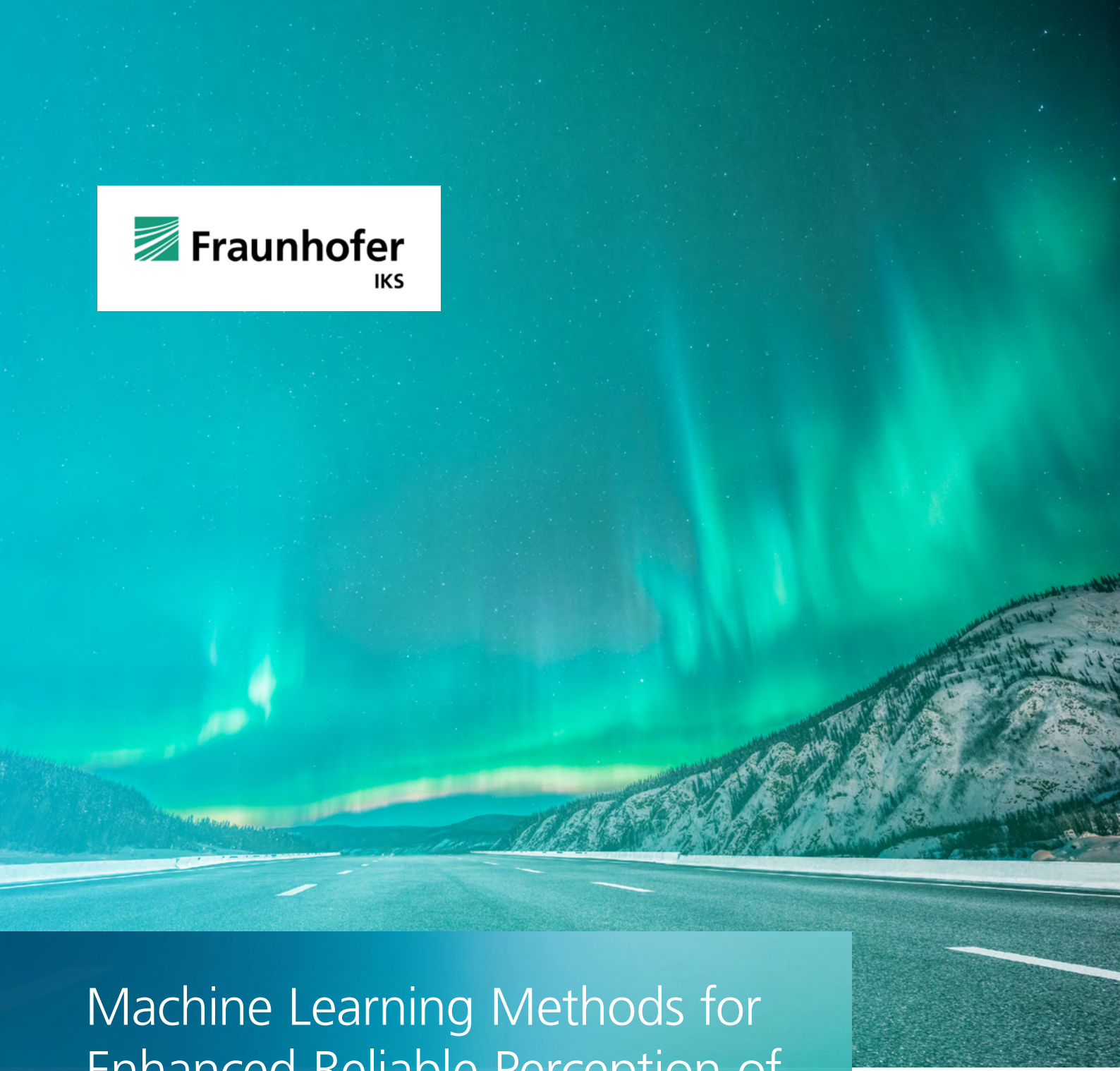

Fraunhofer IKS

Machine Learning Methods for Enhanced Reliable Perception of Autonomous Systems

M. Henne, J. Gansloser, A. Schwaiger, G. Weiss

ADA LOVELACE CENTER

iks.fraunhofer.de

Enhancing the reliable perception of autonomous systems

In our modern life, automated systems are already omnipresent. The latest advances in machine learning (ML) help with increasing automation and the fast-paced progression towards autonomous systems. However, as such methods are not inherently trustworthy and are being introduced into safety-critical systems, additional means are needed. In autonomous driving, for example, we can derive the main challenges when introducing ML in the form of deep neural networks (DNNs) for vehicle perception.

DNNs are overconfident in their predictions and assume high confidence scores in the wrong situations. To counteract this, we have introduced several techniques to estimate the uncertainty of the results of DNNs. In addition, we present what are known as out-of-distribution detection methods that identify unknown concepts that have not been learned beforehand,thus helping to avoid making wrong decisions.

For the task of reliably detecting objects in 2D and 3D, we will outline further methods. To apply ML in the perception pipeline of autonomous systems, we propose using the supplementary information from these methods for more reliable decision-making. Our evaluations with respect to safety-related metrics show the potential of this approach. Moreover, we have applied these enhanced ML methods and newly developed ones to the autonomous driving use case. In variable environmental conditions, such as road scenarios, light, or weather, we have been able to enhance the reliability of perception in automated driving systems.

Our ongoing and future research is on further evaluating and improving the trustworthiness of ML methods to use them safely and to a high level of performance in various types of autonomous systems, ranging from vehicles to autonomous mobile robots, to medical devices.

Contents

1
Introduction

Advances in machine learning and high-performance computing during the last 10 years have led to a huge increase in available methods for improving perception in autonomous systems. These new approaches outperform many conventional, previously used methods on many specific tasks, such as 2D/3D object detection ([78], [60], [80], [101]), 3D depth estimation, image recognition ([45]), and semantic segmentation ([2]) by a large margin. Deep neural networks have turned out to be one of the biggest contributors to this new wave of innovation due to their ability to solve highly complex problems with a high degree of accuracy and are part of every software stack for autonomous driving vehicles today. However, with great power comes great responsibility and a desire to better interpret and quantify the results given by these networks. It has been shown multiple times ([27], [20]) that deep learning models often lack in giving robust and calibrated predictions, making it hard to assess the uncertainties of the outputs. Another important aspect for providing robustness is to monitor and reason about the predictions at test time as well as analyze if new input is far removed from the samples in the training distribution. Calibrated confidence estimates in conjunction with out-of-distribution awareness integrated in safety-critical applications like autonomous driving can provide valuable additional information with respect to situational awareness and can reduce the risk of hazards resulting from functional insufficiencies by decreasing the space of unknown unsafe scenarios, which is a critical part of the safety of the intended functionality (SOTIF ISO/PAS 21448).

In this technical white paper, new methods for quantifying uncertainty in deep neural networks related to perception tasks as well as monitoring system and out-of-distribution approaches will be reviewed, developed and evaluated. Insights and context with respect to different perception tasks, such as object detection, will also be provided and linked to the approaches for estimating uncertainty. The main focus will be on safety-critical applications where these methods provide crucial knowledge to systems for avoiding high-risk behavior and increasing overall safety.

1.1
The Perception Problem

Accurate perception of the surroundings is a key part of many autonomous systems where an agent is required to interact with the environment. Some applications are in a controlled setting where several parameters of the perceived world can be adjusted to meet the requirements for the system to work appropriately.

Fig. 1:
Challenges of automotive sensors in weather conditions (left, [103]) and examples of perception challenges (right, [104])

However, even though these applications can also benefit from more robust predictions with uncertainty quantification, the major target cases for these methods are open-world environments. For these applications, an enormous number of configurations of input data is possible and likely to occur when the system is in its active state. As the inputs for perception modules mostly consist of very high-dimensional sensor data, such as lidar point clouds or (stereo) camera images, and the applied deep learning models also have a large amount of trainable parameters, formal verification that guarantees correct predictions is almost impossible, even for constrained subproblems, due to the huge state space for both inputs and parameters.

2
Problems and Solutions

There are many aspects to consider in order to make machine learning-based systems safe. This section provides an outline and brief summary of the challenges and topics to consider in designing safe systems using ML components.

2.1
Choice of Dataset

The training and test datasets need to be representative for the chosen area of deployment. This requires a thorough understanding of the target domain. It is important to define under which conditions the ML system should operate, e.g., what are the expected objects in the scene or what are the weather conditions. The main challenge here is how to measure and quantify suitable datasets. Metrics include coverage, relevance, equivalence of cases/situations, and coverage of positive and negative examples. Following standard best practices, it is important to train the model independent of the test dataset, e.g., no hyperparameter tuning should be done on test data. Another difficulty is how to ensure that the training data covers the semantics of the data the model will see during deployment. To mitigate the negative effect of distributional shift, the training dataset can be continuously improved by gathering data even after deployment.

How can we capture every relevant critical case required for the expected functionality of the model and how can this be assured? One straightforward approach to this is to acquire large amounts of verification data. This can be done by:

- Augmenting data, e.g., using synthetic data (especially for rare corner cases).
- Analyzing data and finding missing examples, e.g., by searching in a latent space learned by autoencoders.
- An iterative development process – starting in simulation, testing in field studies, etc. – and assessing weaknesses during this process.

Another important aspect is how to verify the quality and correctness of annotations. To consider are:

- The type of annotation: How coarse, e.g., dynamic vs. static objects, hierarchy and/or attributes of objects
- The annotator qualification (especially in the medical domain)
- The dependence of DNNs on labeling quality

2.2
Unexpected Behavior of ML Methods

It is currently not possible to verify deep neural networks and make sure that their behavior is as expected. It is difficult to prove that the network actually learns the semantics of the problem and has a real understanding of the system it should model. Additionally, it is important that a model is robust, i.e., small perturbations in the input should not change the predictions drastically. Another important step towards safer systems is that a network should know when it cannot make reliable predictions. This is related to OOD detection, where the system can, e.g., ask for human help when an input is encountered that is different from the learned concepts. The following list shows concepts that help make ML models more robust and safe. See [94] for a complete list of safety concerns.

Uncertainty quantification learns reliable calibrated uncertainties for each input to determine how sure the model is that its prediction is correct.

Robustness against adversarial attacks changes the learning process of a model such that small – for humans often unrecognizable – changes in the input do not change the prediction. Such adversarial attacks can easily fool an ML system and deployed systems can be easily exploited, potentially resulting in catastrophic failures.

Continuous improvement by gathering data even after deployment. A straightforward approach here is to use semi-supervised learning, where a small amount of labeled data and large amounts of unlabeled data are used for training.

Redundancy and ensembles, with redundancy being applied at many stages of an ML pipeline, e.g., using redundant sensors or different neural network architectures. An often-used approach to improve the performance of an ML model is to train multiple models independently with different initializations.

Out-of-distribution detection is a mechanism to detect novel inputs that have different semantics than those contained in the training dataset. Detecting these is crucial, since standard neural networks often misclassify OOD samples with high confidence. We provide an overview of current OOD methods in Section 4.

3
Uncertainty Estimation

Knowing when a neural network is uncertain is crucial in safety-critical applications. There are two main types of uncertainties that can be modeled. The first one is aleatoric uncertainty (or data uncertainty), which captures the noise inherent in the observations. Examples can be noise from sensors or motion blur due to low sampling frequencies. This kind of uncertainty can be further partitioned into homoscedastic and heteroscedastic uncertainty. Homoscedastic uncertainty assumes constant noise over all input samples – such as inherent measurement uncertainties in sensor data – while heteroscedastic uncertainty assumes non-constant noise depending on each data point – such as blurry camera images during fast motion compared to non-blurry ones. Aleatoric uncertainty cannot be reduced even if more data is available.

In contrast to that, epistemic uncertainty (or model uncertainty) describes the uncertainty about which model to choose for estimating the given data. Standard machine learning approaches simply take the maximum likelihood model to make a point

estimate for new input data. If the full data distribution were available, this would be sufficient as the epistemic uncertainty would be zero. However, this is not the case in the vast majority of problems where a distribution of models better describes the underlying data distribution than a single maximum likelihood model.

3.1 Calibration

For classification, one approach is to directly interpret the scores of the final softmax function as probabilities and calibrate these outputs so that their prediction scores match the probabilities of being the correct classes. This means that predictions with probability ***p*** also have an empirical probability (e.g., accuracy) of ***p***.

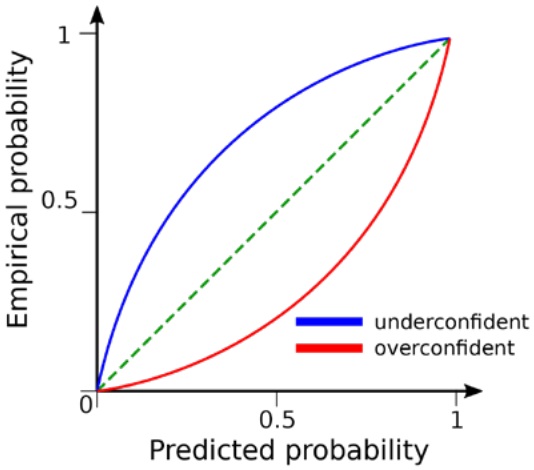

Fig. 2:
Calibration plot

A perfectly calibrated model would correspond to the green dashed line in Figure 2 while over- and underconfident models would correspond to the red and blue curves, respectively. Large increases in model capacity and the complexity of DNNs during the last years, e.g., rising depth, the use of more convolutional filters, and the use of batch normalization, have been found to negatively affect model calibration [27], which often leads to overconfident predictions. For calibrating these DNNs, [27] we can compare old calibration methods like histogram binning or beta calibration and suggest temperature scaling as a new straightforward method for minimizing the expected calibration error. Even though this approach significantly improves model calibration, it still does not present a reliable solution to the uncertainty problem, as the networks are calibrated relative to a certain training dataset. However, for new examples too dissimilar to the training data distribution, the network is not well calibrated anymore, as has been shown in [70].

3.2 Bayesian Neural Networks

The most common methods to account for uncertainty include Bayesian neural networks (BNNs), which usually apply a prior distribution over their parameters θ to compute the posterior distribution over the model parameters given the training data:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

where

$$p(D) = \int_{\theta} p(D|\theta)p(\theta)d\theta$$

is the marginal likelihood of the data/evidence (targets given inputs) with all possible weights marginalized out. Unfortunately, the integral for calculating the marginal probability of the data *p(D)* is analytically intractable in practice, since the number of parameters in a neural network is very large, and we cannot expect the underlying distribution describing the data to be a known closed-form function. For the final prediction during inference time, marginal distribution can be computed as follows:

$$p(\hat{y}_i|x_i, D) = \int_{\theta} p(\hat{y}_i|x_i, D, \theta)p(\theta|D)d\theta$$

Where x_i is a new input and y_i is the corresponding output of the neural network. Due to *p(D)* being intractable, there are two prominent approximation techniques: variational inference and sampling methods. Variational inference methods try to approximate the true posterior over the model parameters with a different distribution from a tractable family (e.g., Gaussian) by finding the parameters of this distribution that minimize the Kullback–Leibler divergence to the true distribution. The problem with variational inference is that it has a very high bias, as we manually choose the form. On the other hand, sampling methods approximate the true distribution by averaging samples drawn from it. One such method is the Markov Chain Monte Carlo (MCMC) algorithm, which constructs a Markov chain with the desired distribution. Even though, in theory, MCMC would lead to a perfect approximation of the true posterior, it is computationally too costly for most DNNs to converge within an acceptable time [34].

3.3 Full Bayesian Approaches

The first works on BNNs ([6], [36]) place an independent Gaussian prior for each weight in a neural network, and then learn the means and variances of these Gaussians using backpropagation. After training, the weight variances can be used to sample diverse networks and obtain diverse predictions as well as the corresponding estimate of epistemic uncertainty:

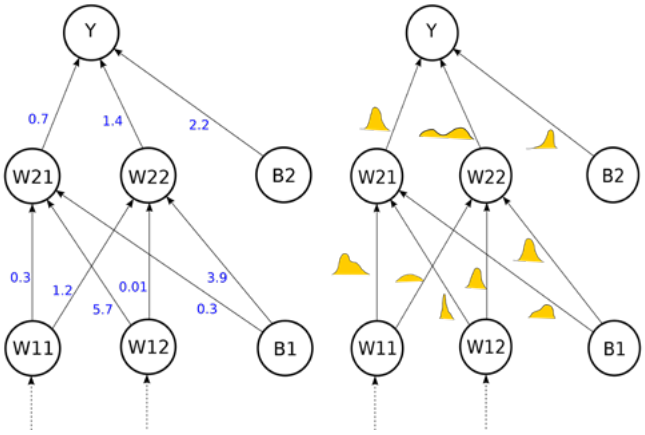

Fig. 3:
Schematic comparison of the weights of deep neural networks and Bayesian neural networks

A major drawback of this approach is that these networks require double the amount of parameters, since each weight consists of a mean and a variance instead of a single value.

3.4 Monte Carlo Dropout

A more recent theoretical finding provides a Bayesian interpretation of the regularization technique known as "dropout" [20]. The argument is that dropout could be used to perform variational approximations of a BNN with a Bernoulli distribution prior from which Monte Carlo sampling is done. In practice, this finding provides an easy way to turn a conventional DNN into a BNN by simply applying dropout during training and testing time. The empirical predictive mean and variance are calculated from multiple stochastic forward passes, where each forward pass can be seen as sampling from a posterior distribution over the network weights. Monte Carlo dropout is easy to implement and can be easily adapted to existing network architectures. Due to its simplicity, high scalability, and good generalization performance, this approach is widely used to tackle the problem of deriving reliable uncertainty estimations of DNNs. Monte Carlo dropout can be interpreted as a form of ensembles with shared network parameters or, alternatively, as approximate Bayesian inference [20].

3.5 Deep Ensembles

Ensembles of DNNs, i.e., deep ensembles, is a well-known method to improve prediction accuracy. However, deep ensembles can also be used for uncertainty estimation [47]. Take a number of randomly initialized neural networks that are trained independently on the same training data. To compute the predictive distribution, the individual prediction probabilities of all neural networks in the ensemble are averaged. Additionally, [47] proposes to use proper scoring functions as loss functions and adversarial training to smooth the predictive distributions.

3.6 Variance Propagation

Another approach for estimating epistemic uncertainty is to use noise injection at inference time to quantify trust in the model in a single shot [72]. It treats noise injected in the neural network as errors in the activation values and applies error propagation, as is done in other sciences like physics. The advantage is that it provides comparable results to Monte Carlo dropout while only requiring a single forward pass.

3.7 Output Distribution Modeling

Another family of approaches to modeling uncertainty in DNNs assumes a certain probability distribution over the network outputs and uses the network output layers to directly predict parameters for such a distribution (e.g., Gaussian). Unlike Bayesian neural networks, these models only predict point estimates, as no marginalization over weights is done. However, by directly modeling the output as a distribution, it is

The most prominent example of directly modeling the output as a distribution is the application of the softmax function, which is equivalent to a multinomial mass function on the final layer of the neural network for classification tasks. For regression tasks, [42]

proposes to model the output as Gaussian-distributed by letting the model predict the mean $\mu = f(x)$ and variance $\sigma^2(x)$. By reformulating the mean-squared error loss function, the heteroscedastic aleatoric uncertainty can be directly learned from data:

$$\hat{y} \sim \mathcal{N} f(x, \theta), \sigma^2(x, \theta))$$
$$L_{reg}(x, \theta) = \frac{1}{2\sigma(x, \theta)^2} ||y - f(x, \theta)||_2^2 + \frac{1}{2} \log \sigma(x, \theta)^2$$

where x is the input and $f(x, \theta)$ the mean prediction of the neural network for the Gaussian. Moreover, y is the regression target and $\sigma(x, \theta)$ is the estimated variance of the output.

3.8 Higher-Order Conjugate Priors

In addition to directly predicting the output probability distributions, several works propose to estimate their higher-order conjugate priors. Prior networks [61] explicitly distinguish distributional uncertainty (is there a mismatch between input data and training distribution) from aleatoric (is the data difficult to classify) and epistemic (how well does the model fit the data) uncertainty. Distributional uncertainty is modeled by parameterizing a prior distribution over predictive distributions. For classification, this is done by using a Dirichlet distribution over the categorical distribution of the classes. The parameters of the Dirichlet distribution are trained using a multitask loss that consists of minimizing the KL divergence between a sharp Dirichlet distribution and the model for in-domain data as well as a flat Dirichlet distribution and the model for out-of-domain data.

We want to highlight here that training a prior network still requires having access to some out-of-distribution data. The authors propose to generate synthetic data at the boundary of the in-domain region.

Evidential deep learning [84] is inspired by the Dempster–Shafer theory and represents a sampling-free approach.

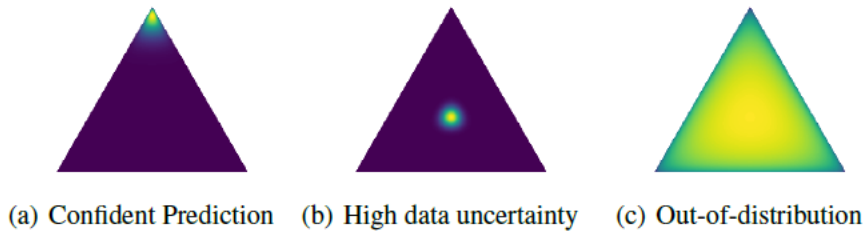

Fig. 4:
Desired behaviors of a distribution over distributions, [61]

Similar to prior networks, for classification tasks, the parameters of a Dirichlet distribution are learned, from which the total evidence for each of the classes and the epistemic uncertainty regarding the prediction as a whole can be calculated. The authors also conducted some experiments regarding out-of-distribution detection and showed that their method generally assigned higher uncertainties to out-of-distribution inputs.

3.9 Ensemble Distillation

A major problem for epistemic uncertainty estimation approaches is the need for sampling in the most prominent ensembling methods such as Monte Carlo dropout or deep ensembles, as described above. This is especially problematic for real-time applications like autonomous driving as the inference speed is reduced by a factor corresponding to the number of sampled used for estimating the posterior. There are approaches to distill an ensemble of models into a single model to yield the mean predictions of the ensemble ([37], [44]). However, while these models perform well on collapsing an ensemble of conditional distributions into a single point estimate conditional distribution over classes, they completely discard the information about the diversity of the models. As a result, epistemic uncertainty can no longer be estimated. In contrast, [62] defines a novel task ensemble distribution distillation, where the goal is to distill the whole distribution of an ensemble instead of just a point estimate. For the classification task, they distill an implicit distribution over distributions (ensemble of predictions over classes) into an explicit distribution over distributions (Dirichlet distribution over predictions over classes) by using a single prior network model, as in [61]. [58] generalizes this approach to allow for freedom in choosing the parameterization of the output distribution, which makes it possible to also distill models for other tasks than classification, such as regression.

4 Out-of-Distribution Detection

Out-of-Distribution (OOD) methods detect inputs that are conceptually different from the training dataset. In the real world, a deployed model is often confronted with novel data that was not present in the training dataset. This can be samples from new classes or even samples that contain completely new concepts.

To formalize this, let P_D and P_Q be two distinct data distributions, where P_D is the in-distribution and P_Q is the out-distribution. We train a model on a dataset sampled from P_D. The task of OOD detection is to determine if an input sample is from the in-distribution or not when the model is queried on a testing dataset consisting of in- and out-of-distribution samples from P_{DxQ}.

When confronted with unknown data, a machine learning model cannot make correct and reliable predictions. Neural networks tend to make overconfidently wrong predictions for OOD inputs [27, 32]. To use machine learning models in safety-critical applications, it is crucial to detect these OOD inputs. When a machine learning model knows when it is confronted with unknown input data, it can, e.g., ask for human input or choose a more reliable safety path.

An important characteristic of OOD methods is whether OOD data is required during training. Using OOD data for training imposes a strong requirement on the problem, since it is often difficult to acquire sufficient suitable OOD data. Since any data not present in the training dataset could be OOD, using a specific dataset as OOD training data could induce a bias in the learning process.

We note here that many of the methods discussed in the literature are proposed under different names or from different viewpoints, e.g., open set recognition, anomaly detection, novelty detection, rejection, zero-shot learning, uncertainty estimation, and all are closely related or equal to out-of-distribution detection. We will review some of the most popular and best-performing methods in the following section, focusing on

the task of multiclass classification.

4.1 Softmax Scores

A straightforward approach is to interpret the softmax scores of a classifier as probabilities, where

$$y_i = \frac{\exp f_i(x)}{\sum_{j=1}^{C} \exp f_j(x)},$$

and $f_i(x)$ are the logits for class i and input x for C classes. These softmax scores are then used to detect OOD inputs by simple thresholding, as shown in the equation below.

$$g(y;\delta) = \begin{cases} 1 & \text{if } \max_i y_i \leq \delta \\ 0 & \text{else} \end{cases}$$

It has been shown that ID data tends to have larger softmax scores than OOD data. This fact is used to establish a simple baseline on OOD detection by thresholding the softmax score [32]. However, softmax probabilities are often overconfident and not well calibrated [20].

Uncertainty estimation: Many of the presented methods use uncertainty estimates to detect OOD inputs, where inputs with high uncertainty are flagged as OOD [83].

A Bayesian approach to OOD using uncertainty estimation is proposed in [26]. A neural linear model (NLM) is trained, where the neural network features are learned by augmenting the dataset with generated OOD samples that lie on the edge of the decision boundary. These samples are generated using a normalizing flow generative model. It has been shown that the NLM uncertainties are comparable to uncertainties from Gaussian process classification.

The ODIN detector [54] uses temperature scaling and adversarial learning to make the softmax probabilities a better score for distinguishing ID and OOD samples. In the original paper, the temperature scaling parameter is trained on a OOD validation dataset. An extension of it, generalized ODIN [39], proposes to explicitly model the in-distribution by decomposing the class posterior probability. It has been shown that this method achieves competitive results without the need for OOD data during training.

4.2 Outlier Exposure

In outlier exposure [33], large amounts of OOD data are used to train an OOD detector to discriminate between P_D and P_Q. It has been shown that this model generalizes well to unseen disjoint OOD data. The representations are learned by an auxiliary classification task, where the loss function is of the form:

$$\mathcal{L} = \mathcal{L}_{\text{in}}(X_{\text{in}}, Y_{\text{in}}) + \mathcal{L}_{\text{out}}(X_{\text{out}}),$$

where L_{out} is the cross-entropy loss between the predictive distribution and a uniform

distribution. The L_{in} term can be chosen depending on the auxiliary task. In the case of a classification auxiliary task, the L_{in} can be the cross-entropy. For density estimation, L_{in} can be a ranking loss based on the likelihood of ID and OOD samples. Related to outlier exposure, the authors in [5] introduce guaranteed upper bounds around the l_∞-ball of an input sample. This ensures not only low confidence of OOD samples but also low confidence of samples that are close to it, which makes the OOD detection more robust against adversarial attacks.

4.3 Density Estimation

A straightforward approach to OOD detection is to estimate the probability density of the in-distribution dataset and compare the likelihoods of predictions to determine if a sample is OOD. Likelihood-based generative models, like PixelCNN [82, 91, 69] and GLOW [43], estimate the probability density of the training data in an unsupervised fashion and allow for exact log-likelihood computation. However, despite their good results in terms of likelihood estimation, when applied to OOD detection, these models still perform worse than other state-of-the-art OOD detectors [13, 67, 85].

4.4 Contrastive Learning for ODD Detection

Contrastive learning aims to learn a meaningful embedding space by contrasting positive samples with augmented samples and negative samples. Usually, the loss is computed by considering each sample and its sets of positive and negative samples, where the positive samples are augmented version of the original sample. The contrastive loss enforces that the current sample and all all positive samples in the embedding space are pulled together, while at the same time pushing them apart from the set of negative samples.

This learned embedding space can then be used to improve the performance of various downstream tasks. In the case of OOD detection [95], the embedding is learned by combining the contrastive loss and a classification loss. The contrastive loss uses augmented input samples and learns to pull the augmented images based on the same image together and push all other images away. Note that even samples from the same class are pushed apart. To perform OOD detection, the embedding space is used to estimate a Gaussian distribution for each class. To determine if a sample is OOD, the maximum probability is calculated across all classes. All samples that have a maximum probability below a threshold are OOD.

Another approach using contrastive learning is employing specific distribution-shifting transformations for generating negative samples [88]. The authors show that these augmentations improve OOD performance, a combination of cosine similarity to all training samples and the norm of the latent representation is used. An alternative score is proposed that additionally incorporates the shifting transformations.

4.5 Selective Prediciton and Rejection

Another approach is rejection [3, 14, 21], where a machine learning system can chose to reject an input in favor of making wrong predictions. During training, a model is allowed to reject samples for a cost instead of making wrong predictions that would yield a larger loss. More sophisticated methods train a separate rejection function and

learn an optimal threshold parameter or other performance guarantees. One can argue that rejection does not solve the general OOD detection problem, since it only safeguards against inputs that would be wrongly recognized and not against unknown inputs.

4.6 Open Set Recognition

Open set recognition (OSR) is closely related to OOD detection. Like OOD detection, OSR solves the problem of unknown classes during test time. However, OSR has a more defined problem formulation [9, 23] compared to OOD detection and some well-known OOD methods do not satisfy the properties imposed by OSR. The essential properties for OSR are bounding the open space risk and balancing it with the empirical risk. Bounding the open space risk minimizes the volume of space represented by a recognition function, which determines known positive samples. An important factor here is still to allow the classifier to generalize well enough. One approach to OSR using deep learning was proposed in [4], where the distances between the activations of the penultimate layers of an input sample and the mean activations over all correctly classified training samples are used to determine if a sample is OOD.

5 Object Detection

Like many fields utilizing machine learning methods, most research efforts focus on improving model performance on some well-established benchmark datasets, like COCO [57], PASCAL VOC [18] or KITTI (mostly for 3D object detection) [22]. Nevertheless, even though uncertainty estimation methods are more explored in image classification, they are also gaining increasing popularity in the field of object detection, as there is a serious need for reliable predictions in many applications. The perception pipeline of an autonomous system can involve 2D or/and 3D object detection models in multiple variations and combinations. Since the methods for extracting features and uncertainties are different in 2D than in 3D, they are discussed in separate sections.

5.1 2D Object Detection

2D object detection is an essential part of perception tasks in many pipelines. The goal is to detect the bounding boxes covering objects in images accurately and to classify them correctly. RGB images provide context-rich representation of the environment and are, therefore, used in many applications, such autonomous driving and robotics. This section discusses general state-of-the-art methods for 2D object detection in depth. Additionally, several uncertainty estimation methods in 2D object detection are highlighted.

5.1.1 Methods

This section briefly discusses current state-of-the-art 2D object detection methods. An object detector is usually composed of two parts: a backbone, which is mostly a CNN pretrained on a large classification dataset, such as ImageNet [15]; and a head, which predicts classes and bounding boxes for objects. The backbone network could be VGG [87], ResNet [30], DenseNet [40], EfficientNet [89], CSPNet [92] for large or medium-sized computing resources, or smaller networks, such as MobileNet [38] or SqueezeNet [41], for applications on low-power devices. For the head, approaches are usually

separated into two categories: one-stage and two-stage object detectors. The most popular two-stage detectors are the family of R-CNN models, including fast R-CNN [24], faster R-CNN [80], and Libra R-CNN [71]. Two-stage detectors use a region proposal network to generate regions of interests in the first stage and utilize these proposals in the classification and bounding box regression part. These models achieve the highest accuracies but are typically much slower than one-stage detectors. As for one-stage detectors, the most representative models are SSD [60], RetinaNet [56], EfficientDet [90] and YOLO ([79], [76], [77], [7]). One-stage detectors directly predict the classification and regression output for a given configuration of predefined locations. Unlike two-stage detectors, they do not utilize a second regional proposal network to find regions of interest, instead they directly map the input to the number of defined output nodes.

Most one-stage and two-stage detectors are anchor-based, meaning they use a predefined set of boxes for each region of interest (in one-stage detectors, mostly grid cells of feature maps; in two-stage detectors, the output of an RPN) with a variety of aspect ratios and scales to match them to the ground truth with the highest respective intersection over union (IoU). However, anchor-free one-stage detectors like CenterNet [16]or CornerNet [49] have also recently gained some popularity. While two-stage methods tend to be more accurate, one-stage detectors are much faster, reaching inference time rates that are suitable for real-time applications. Recently developed object detectors often contain some layers between backbone and head, which are usually used to aggregate feature maps from different stages.

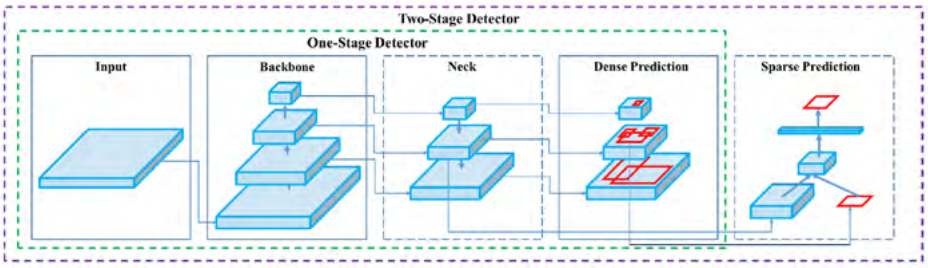

Fig. 5:
Object detection pipeline for different models [7]

These layers are mostly composed of several bottom-up and top-down paths, and are usually referred to as the neck. Models containing such a mechanism include feature pyramid networks (FPNs) [55], path aggregation networks (PANs) [59] and bipolar feature pyramid networks (BiFPNs) [90].

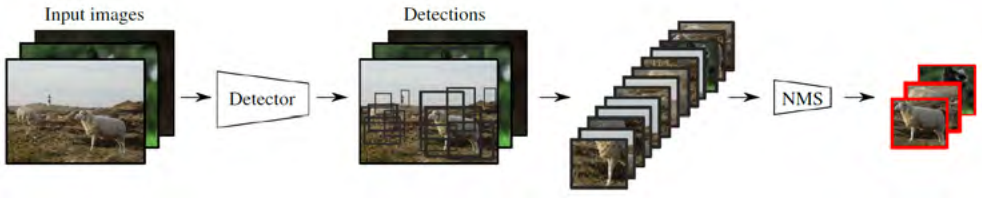

Fig. 6:
Non-maximum suppression

Additionally, almost all detectors apply some sort of suppression algorithms, such as non-maximum suppression or soft non-maximum suppression [8], at the end of each prediction to suppress redundant outputs; see Figure 6.

The loss function for training 2D object detectors is primarily the negative log-likelihood per class for classification and the Huber loss for bounding box regression. Neverthel-

ess, recently, the idea of directly training on an IoU-related objective [100, 81] with the ground truth has gained popularity, as models trained with this kind of regime have improved in accuracy.

Recently, another very distinct but also highly promising approach to 2D object detection using transformers has gained a lot of attention and popularity. Detection transformers (DETR, [11]) reformulate object detection as a direct set prediction problem. They use learned object queries to reason about the relations of the object together with the global image context to directly output a final set of predictions. In contrast to other object detection methods, there is no need for hand-crafted components, such as anchor grids or non-maximum suppression. The current drawback of this approach is the high computational cost of calculating the attention matrix over a two-dimensional feature map representation. However, while other methods at the time of writing still outperform transformer-based methods, there are good chances that transformer-based end-to-end approaches could become state-of-the-art in the future.

5.1.2 Uncertainty in 2D Object Detection

In 2D object detection, there are, apart from epistemic and aleatoric uncertainties, two additional types of quantifiable uncertainties: label and spatial uncertainty. The uncertainty about the label is equivalent to the case in classification problems, where it describes the confidence a prediction has that a given detection has a certain class. Spatial uncertainty, on the other hand, represents the uncertainty associated with the position of the detected bounding box, which is higher for boxes that are expected to deviate more from the ground truth and vice versa.

In order to estimate epistemic uncertainty, similar approaches to standard classification and regression tasks are applied in 2D object detection. In [29, 65], the authors applied Monte Carlo dropout, as described in Section 3, to object detection. The uncertainty is then estimated as a sample statistic from spatially correlated predictions. Following their work, [66] examined the effect of several association and box-merging strategies on the quality of the uncertainty estimates. It was shown that the black box method from [65], which uses NMS, gives weakly correlated results with respect to the bounding box uncertainty. The main reason for this is the removal of redundant information due to the NMS algorithm.

For modeling aleatoric uncertainty for the bounding box regression tasks, various deviations of output distribution modeling approaches as described in Section 3 are used. Below is a recap of the most common formulation for regression tasks:

$$\hat{y} \sim \mathcal{N} f(x,\theta), \sigma^2(x,\theta))$$
$$L_{reg}(x,\theta) = \frac{1}{2\sigma(x,\theta)^2}||y - f(x,\theta)||_2^2 + \frac{1}{2}\log\sigma(x,\theta)^2$$

By placing a Gaussian distribution over the predictions of the bounding box coordinates with the direct prediction of the diagonal elements of the covariance matrix, this formulation was directly used by [50] as the loss for the bounding box regression in 2D object detection. [31] incorporates bounding box variances to calculate the Kullback–Leibler divergence between the predicted Gaussian distribution and the ground truth box that is also modeled as a Gaussian, with $\sigma = 0$, which results in a Dirac delta distribution.

Metrics: The most commonly used metric to evaluate the performance of 2D detectors, introduced in [18], is mean average precision (mAP), which measures the average of the maximum precision values at different recall values over all classes and

thresholds. However, average, precision-based methods rely on fixed IoU thresholds to determine positive and negative detections, causing an additional dependency on the threshold hyperparameter. Furthermore, the label score is taken as detection-ranking evidence instead of a spatial quality measure, leading to suboptimal detection assignments. Another problem with mAP is the metric's lack of uncertainty awareness , as confidence only plays a role in determining if a sample is positive, transforming some kind of confidence value attached to each detection into a binary decision without considering the exact value itself. For example, if the threshold of a detection is defined as positive for a label score above 0.5 and negative below, a prediction of 0.51 confidence will have the same impact on the score as one with 0.99, even though the values differ greatly. Another similar problem is the definition of a true positive, which also requires hand-selecting the threshold.

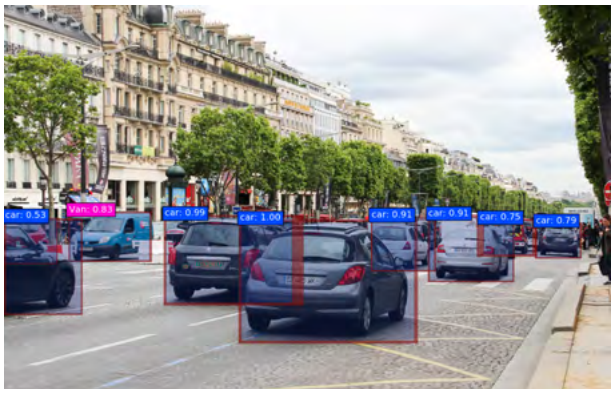

Fig. 7:
Prediction on an ensemble of seven SSD object detectors, with red showing the standard deviation for the bounding box locations

To avoid these problems, a new uncertainty-aware metric was introduced by [28], called probabilistic detection quality (PDQ). This metric has a more continuous approach to defining the quality of the predictions. Compared to mAP, it does not need a hand-chosen threshold parameter. Instead, the metric consists of two components: *label quality* and *spatial quality*.

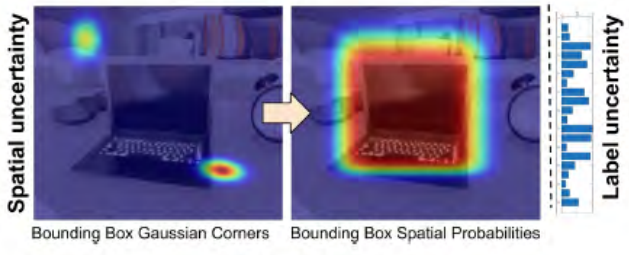

Fig. 8:
Object locations are represented as probabilistic bounding boxes, where corners are modeled as 2D Gaussians, giving the uncertainty for each pixel [28]

Label quality measures how effectively a prediction identifies the correct class label. It is defined as the probability $Q_{\mathcal{L}}$ estimated by the detection model for the ground truth class of the object. This way, the entire probability mass associated with the correct label is considered in a continuous way. Spatial quality Q_S is composed of a foreground loss $L_{\mathcal{F}}$ and a background loss $l_{\mathcal{B}}$, and measures how well a detection captures the ground truth spatially by taking into account the spatial probabilities for individual pixels expressed by the detection model; see Figure 8.

5.2
3D Object Detection

For the task of 3D object detection, there is a large variety of approaches to achieving state-of-the art performance on common benchmarks, such as KITTI [22] or the recently published NuScenes dataset [10]. The reason for the greater diversity of methods in comparison to 2D object detection is that there exist multiple data formats to capture the environment – unlike with 2D, where images are the only source of information. Most methods work with data from a single sensor or a combination of multiple sensors. The most commonly used sensors are lidar, mono/stereo/RGB-D camera(s) and radar. The number of classes available for 3D object detection is also considerably lower compared to 2D object detection. The classes available in 3D datasets are mostly limited to a subset of classes relevant to use in autonomous vehicles. The most common labels are cars, cyclists and pedestrians.

5.2.1
Methods

As mentioned before, the variety of methods that can be used is very large. There are many ways the data from different sensors can be represented, processed and fused with a large number of different network architectures. In this section, we detail some of the most important approaches and present previous works applying each strategy. There are multiple representations of data for 3D object detection. Many methods use 2D representations, which can be directly captured by standard and RGB-D cameras, or extracted from lidar or radar data by applying transformation methods. A different approach is to directly process 3D point cloud data. Other methods fuse together multiple representations from one or multiple modalities to extract more meaningful features.

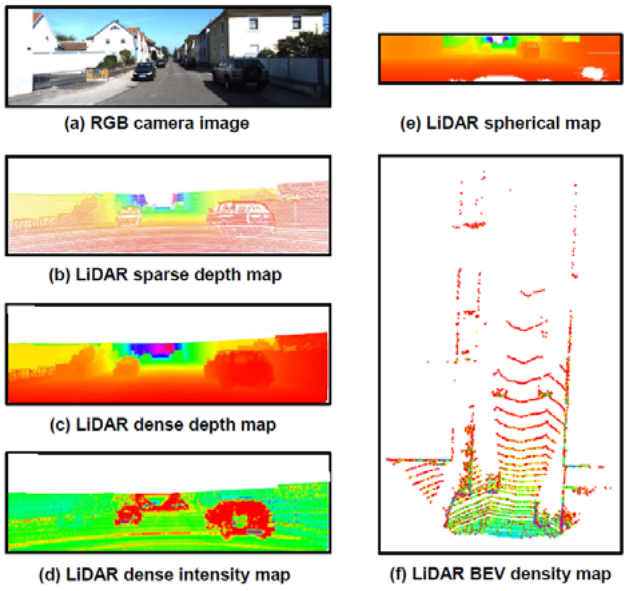

Fig. 9:
Different 2D views, [19]

Methods Based on Front-View Images: Methods using 2D representations can be divided into two classes: those based on a front view and those based on a bird's eye view. Front view representations can be RGB images, sparse/dense depth maps, intensity maps or spherical maps; see Figure 9. For 3D object detection, these maps have the advantage of representing each 3D point in a dense and compact manner, which can also be helpful for tasks such as point cloud segmentation [96]. [52] converts point clouds into front-view 2D maps in order to apply 2D detectors to localize objects

in the front-view images. The advantage of front-view maps is that it can be directly fused with camera images if their sizes are the same. Nevertheless, such representations often leave many pixels empty. To deal with this issue, some methods have been proposed to up-sample sparse feature maps, such as nearest neighbors [1] or bilateral filters [73]. More recently, [64] has used a multichannel range view image from an RGB-D camera as the input and has applied a fully convolutional neural network to predict a multimodal distribution of 3D boxes for each point. Another very interesting, new branch of approaches converts RGB images into a pseudo-lidar [93, 99] representation by estimating the image depth and applying the more powerful bird's-eye view methods on these generated point clouds.

Methods Based on Bird's-Eye View: In contrast to front-view representations, the bird's-eye view preserves the length and width of objects and avoids occlusion problems. It also provides the position of objects directly on the ground plane, making the localization task easier. MV3D [12] is the first method to convert point cloud data into a bird's-eye view representation. This is done by discretizing the projected point cloud into a 2D grid on the ground plane. To capture more detailed height information, the point cloud is divided into a fixed number of slices in the vertical direction to obtain height maps. These maps are then concatenated with intensity and density maps to obtain multichannel features. ComplexYOLO [86] uses a YOLO network [79] together with an angle encoding approach to increase runtime and orientation performance. Pixor [98] is a fast, single-stage, proposal-free detector that is designed to efficiently make use of specific height-encoded bird's-eye view data. Another approach called Pointpillars [48] uses PointNet features [75] to encode a point cloud into a sparse pseudo-image. These features are further processed by a 2D convolutional backbone and an SSD-style detection head for detecting and regressing 3D boxes.

Voxel-Based Methods: Another way of representing input data is to divide the 3D space into equally-shaped 3D voxels and assign the points from a point cloud to the discretized voxels [51, 17, 102]. This representation allows the rich 3D information to be preserved. However, this comes at a high computational cost, as many voxels are empty due to the sparsity of many point clouds. [102] groups point cloud data into voxels, extracts PointNet features by voxel [75] using a voxel feature encoding layer and then converts these features into a dense tensor to be processed using 3D and 2D convolutional networks. The problem of this approach is the expensive 3D convolutions, where the computational effort has a cubic scaling with respect to the voxel resolution. In order to reduce the complexity, [97] uses sparse 3D convolutions [25] in combination with other improvements to drastically reduce inference time for a VoxelNet-based approach. The Pointpillar-based [48] approach described in the bird's-eye view part is very similar to voxel-based methods, with the difference being that the height dimension is skipped entirely, making 3D convolutions unnecessary.

Fusion-Based Methods: The majority of recent works combines camera images with lidar point clouds to extract more meaningful features and improve performance. Many approaches fuse camera and lidar features extracted from 2D convolutions. To accomplish this, these methods project lidar points onto a 2D plane to further process the 2D feature maps from camera and lidar data together through 2D convolutions. In MV3D [12], point cloud data is converted into a front view and a bird's-eye view, and then feature maps extracted from both point cloud maps are fused with an image feature map. Another approach uses 2D object detection to constrain the space for possible objects in 3D and then uses PointNet features [75] in these constrained regions for 3D detection [74]. Other fusion-based approaches cluster and segment 3D lidar points to generate 3D region proposals by using a 2D lidar representation to extract features for fusion ([68], [63]). Other works project lidar points onto the camera plane or RGB images onto the bird's-eye view plane in order to join the features together ([53]) or fuse bird's-eye view features directly with RGB images [46] in conjunction with a regional proposal network for 3D proposals, similar to [80] for 2D.

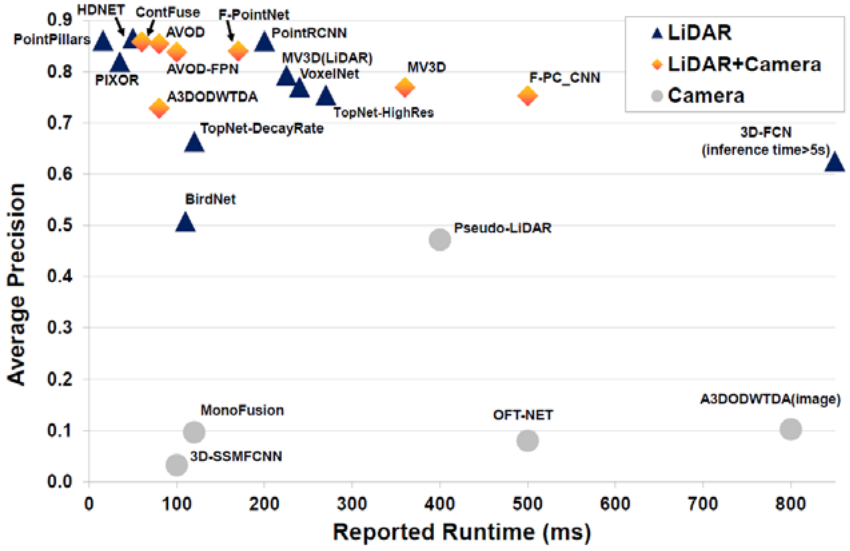

Fig. 10:
Runtime vs. precision comparison of different 3D object detectors on the KITTI bird's-eye view test dataset [19]

6
Application of ML Methods

The aim of the presented ML methods is to assist in ensuring the safety of a perception system for autonomous vehicles. To this end, we investigated the use of uncertainty quantification for object detection, one of the essential perception tasks of autonomous systems. The additional reliability information gained from the uncertainty quantification can be utilized to increase the overall safety of the system [34]. Figure 11 shows the envisioned concept for incorporating the uncertainty information in a perception pipeline.

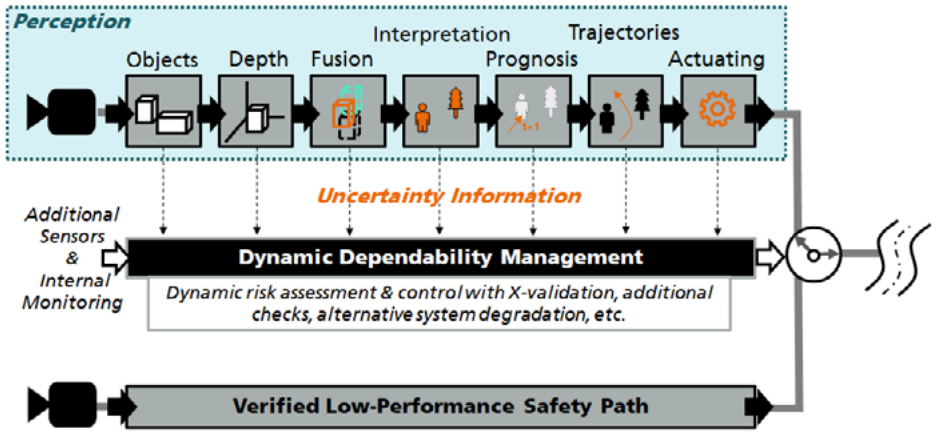

Fig. 11:
The envisioned perception pipeline, integrating reliability information such as uncertainty estimates

The overall perception system is thus split into two separate paths: a high-performance path based on ML methods and a fallback path relying on classical, non–data-driven approaches. The subsystems in the upper high-performance path are extended by additional reliability information, such as uncertainty estimates. At runtime, a dynamic dependability management system takes this reliability information into account and combines it with additional sensor information and other monitoring systems. In each given situation, it dynamically assesses the reliability of the outputs of the high-performance path and, if required, switches to the safety path. The safety path is intended to provide basic functionality to perform minimal-risk maneuvers, e.g., coming to a stop on the hard shoulder of a freeway, ensuring that the vehicle is always in a safe state.

In order to derive results on the general suitability of the methods for assisting in assurance, we conducted different evaluations by benchmarking the underlying feature extractors of the object detector and comparing it to other state-of-the-art approaches with respect to safety metrics.

6.1 Benchmark of Uncertainty Quantification Methods for DNNs on the Task of Image Classification

To find suitable uncertainty estimation approaches that can be considered further for robust object detection, we first performed two benchmarks on the underlying task of image classification. In the first [35], we compared four uncertainty quantification methods for DNNs – Monte Carlo dropout, deep ensembles, evidential deep learning and learned confidence – against the baseline of assuming the outputs of the softmax activations used for classification as confidences. We evaluated their performance across three standard image classification datasets – MNIST, CIFAR-10 and the German Traffic Sign Recognition Benchmark– and two network architectures – a simple six-layer CNN and VGG16. For this, we used evaluation metrics that also take into account safety-related aspects: network calibration, which measures how well confidences are calibrated; and remaining error rate vs. remaining accuracy rate, a metric we introduced to capture the trade-off between performance and safety when discarding inputs based on a confidence threshold.

Figure 12 shows the results of the trade-off between remaining accuracy and remaining error based on different thresholds for the confidence for the six-layer CNN. Deep ensembles thus consistently outperform the other methods, followed by learned confidence. Monte Carlo dropout can improve upon the softmax baseline for CIFAR-10, however, it performs the worst on GTSRB. Evidential deep learning in this benchmark suffered from unstable training, which is why it is excluded from the GTSRB results. In our second benchmark, shown later, we resolved this issue, showing the potential of this method. For VGG-16, we saw similar results for most methods. One exception was Monte Carlo dropout, which showed very little variance in the individual predictions, thus performing very similarly to the softmax baseline. We presumed the reason for this to be overparametrization, i.e., the network had the capacity to learn various redundant paths, which, in turn, were robust – in a negative sense – to the dropout masks at inference. Figure 13 shows the calibration of the individual uncertainty estimation approaches. As expected, softmax is significantly overconfident, while deep ensembles and Monte Carlo dropout are mostly well calibrated. Both sampling-free approaches, evidential deep learning and learned confidence, are highly underconfident – a property beneficial to safety but that severely impacts the performance of the network. To summarize, deep ensembles consistently showed the best results. However, due to the increased computational cost in training and inference, learned confidence as a sampling-free approach may also be of interest in further studies.

In our second benchmark [83], we investigated whether the same uncertainty quantification methods used in the first benchmark were suitable for detecting novel concepts in input images that otherwise would lead to false positives. To this end, for each method, we trained three different architectures—VGG16, SqueezeNet and EfficientNet—on three different datasets—CIFAR-10, German Traffic Sign Recognition Benchmark and NWPW-RESISC45. For the evaluation, we chose an out-of-distribution dataset for each training dataset: CIFAR-100, Belgian Traffic Sign Dataset and a different split for NWPU-RESISC45. We investigated whether discarding predictions based on uncertainty allows for rejecting novel inputs without impacting overall performance too much.

Figure 14 shows some results from the benchmark. Ideally, an uncertainty quantification approach suitable for out-of-distribution detection would minimize the difference between the blue and green curves. It shows that deep ensembles again consistently showed the best results, closely followed by evidential deep learning, which is a sampling-free approach. However, it was also evident that, for a truly reliable novelty detection approach, other, more specific measures are required, as a significant portion of out-of-distribution inputs could not be discarded by any method without greatly impacting overall performance.

The benchmarks were performed on the task of image classification and intended to find suitable approaches to transfer to the downstream task of object detection, for which we developed a specific uncertainty quantification approach. To this end, we decided to further investigate in the direction of deep ensembles how to transfer them to object detection and how to minimize their computational complexity.

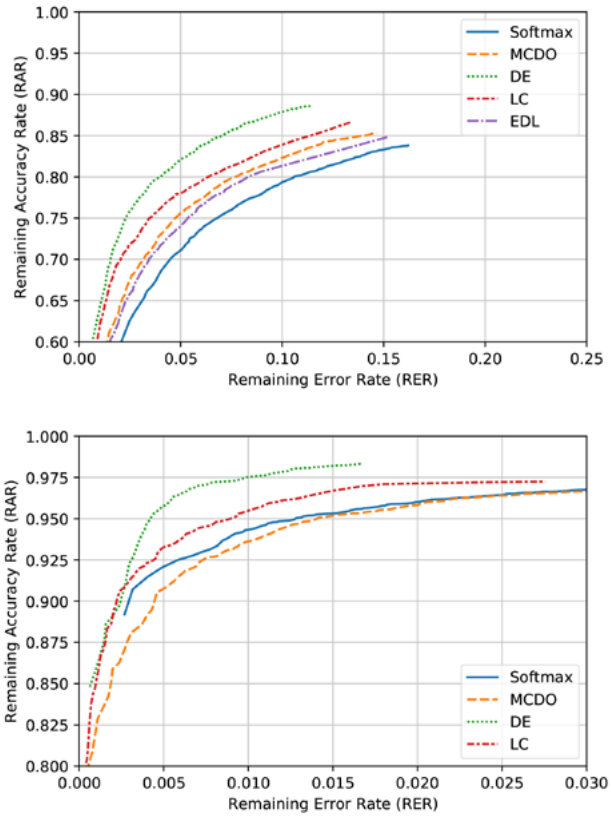

Fig. 12:
Trade-off between remaining accuracy and remaining error for the six-layer CNN with different uncertainty quantification approaches

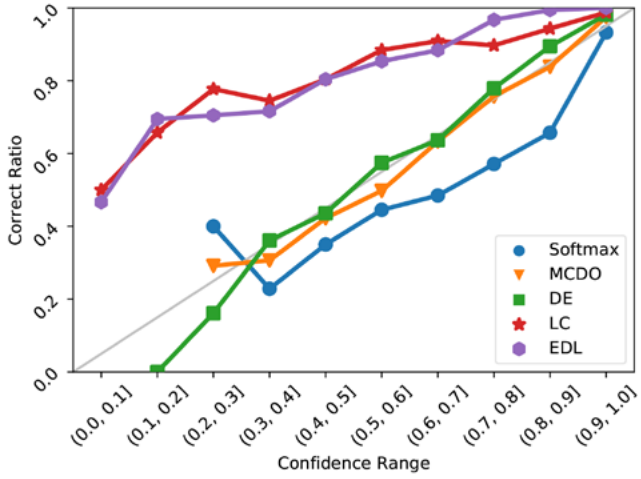

Fig. 14:
Confidence calibration of the six-layer CNN on CIFAR-10 with different uncertainty quantification methods

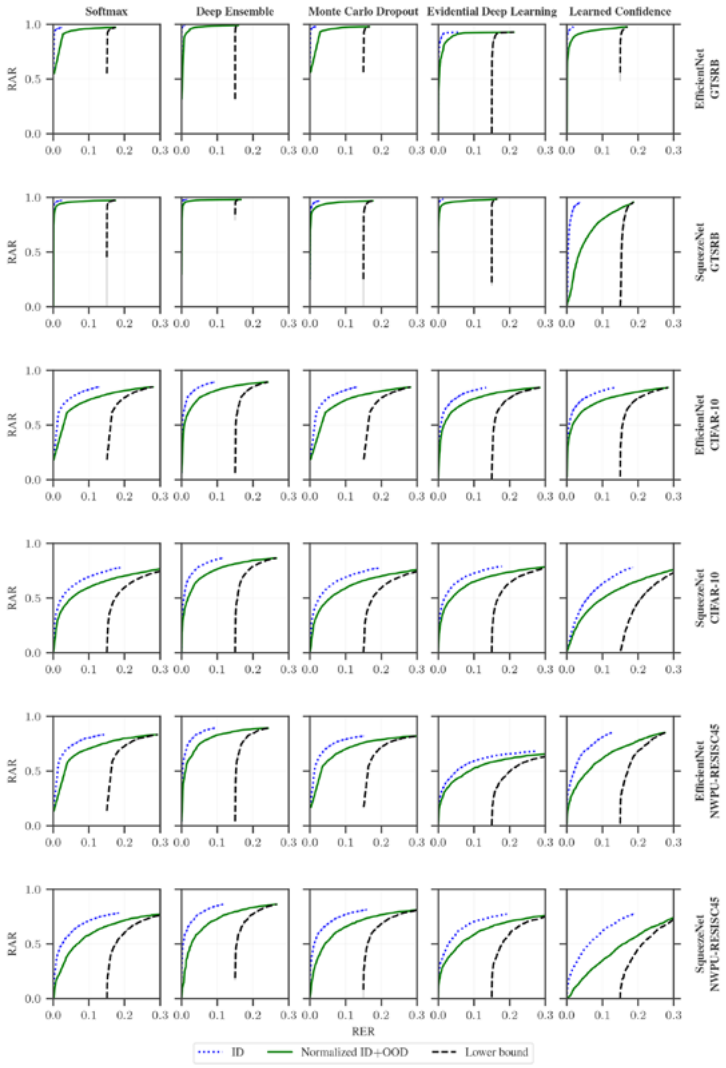

Fig. 13:
Results for the out-of-distribution benchmark, showing remaining error rate (RER) vs. remaining accuracy rate (RAR) for EfficientNet and SqueezeNet on the GTSRB, CIFAR-10 and NWPU-RESISC45 datasets. The plots show the performances first on the ID dataset (blue), then on a dataset consisting of ID and OOD samples (green). The lower bound (black) represents the worst-case scenario, where the network fails to reject any of the OOD samples

7
Outlook

Machine learning is a core technology for making autonomous systems become reality in many application areas. For this, the key is to reliably integrate them into safety-critical systems, such as autonomous cars or collaborative robots.

We investigated current methods and approaches for enhancing the reliability and robustness of ML-based perception. On this basis, we researched supplementary methods to apply them successfully in autonomous systems' architectures. Our evaluations of diverse autonomous driving scenarios show the enhancements in ML-based perception using these approaches. When it comes to changing environmental conditions and road scenes, we achieve more reliable detection rates.

Nevertheless, from a safety perspective, the introduction of ML into safety-critical tasks still requires application-specific solutions. Therefore, our research focuses on developing feasible solutions for real-world systems. We utilize our own building blocks and patterns as well as those provided by others as much as possible, enabling the advantages of ML in the products of today and tomorrow.

8 References

[1] Alireza Asvadi et al. "DepthCN: Vehicle detection using 3D-LIDAR and ConvNet". In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). IEEE. 2017, pp. 1–6.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "SegNet: A deep convolutional encoder–decoder architecture for image segmentation". In: IEEE transactions on pattern analysis and machine intelligence 9.12 (2017), pp. 2481–2495.

[3] Peter L Bartlett and Marten H Wegkamp. "Classification with a reject option using a hinge loss". In: Journal of Machine Learning Research 9. Aug (2008), pp. 1823–1840.

[4] Abhijit Bendale and Terrance E Boult. "Towards open-set deep net . works". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016, pp. 1563–1572.

[5] Julian Bitterwolf, Alexander Meinke, and Matthias Hein. "Certifiably Adversarially Robust Detection of Out-of-Distribution Data". In: Advances in Neural Information Processing Systems 33 (2020).

[6] Charles Blundell et al. "Weight uncertainty in neural networks". In: arXiv preprint arXiv: 1505.05424 (2015).

[7] Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: arXiv preprint arXiv: 2004.10934 (2020).

[8] Navaneeth Bodla et al. "Soft-NMS–improving object detection with one line of code". In: Proceedings of the IEEE international conference on computer vision. 2017, pp. 5561–5569.

[9] Terrance E Boult et al. "Learning and the unknown: Surveying steps toward open-world recognition". In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. 2019, pp. 9801–9807.

[10] Holger Caesar et al. "nuScenes: A multimodal dataset for autonomous driving". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, pp. 11621–11631.

[11] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: arXiv preprint arXiv: 2005.12872 (2020).

[12] Xiaozhi Chen et al. "Multi-view 3D object detection network for autonomous driving". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, pp. 1907–1915.

[13] Hyunsun Choi, Eric Jang and Alexander A. Alemi. "WAIC, but Why? Generative Ensembles for Robust Anomaly Detection". In: arXiv preprint arXiv: 1810.01392 (2018).

[14] Corinna Cortes, Giulia DeSalvo and Mehryar Mohri. "Learning with rejection". In: International Conference on Algorithmic Learning Theory. Springer. 2016.

[15] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: 2009 IEEE conference on computer vision and pattern recognition. EEE. 2009, pp. 248–255.

[16] Kaiwen Duan et al. "CenterNet: Keypoint triplets for object detection". In: Proceedings of the IEEE International Conference on Computer Vision. 2019, pp. 6569–6578.

[17] Martin Engelcke et al. "Vote3deep: Fast object detection in 3D point clouds using efficient convolutional neural networks". In: 2017 IEEE
International Conference on Robotics and Automation (ICRA). IEEE.

2017, pp. 1355–1361.

[18] Mark Everingham et al. "The pascal visual object classes (VOC) challenge". In: International Journal of Computer Vision 88.2 (2010), pp. 303–338.

[19] Di Feng et al. "Deep multi-modal object detection and semantic segmentation for autonomous driving: datasets, methods, and challenges". In: IEEE Transactions on Intelligent Transportation Systems (2020).

[20] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: Proc. ICML 2016. Vol. 48. PMLR, June 2016, pp. 1050–1059.

[21] Yonatan Geifman and Ran El-Yaniv. "Selective classification for deep neural networks". In: Advances in Neural Information Processing Systems. 2017, pp. 4878–4887.

[22] Andreas Geiger, Philip Lenz and Raquel Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE. 2012, pp. 3354–3361.

[23] Chuanxing Geng, Sheng-jun Huang and Songcan Chen. "Recent advances in open-set recognition: a survey". In: IEEE Transactions on Pattern Analysis and Machine Intelligence (2020).

[24] Ross Girshick. "Fast R-CNN". In: Proceedings of the IEEE international conference on computer vision. 2015, pp. 1440–1448.

[25] Benjamin Graham and Laurens van der Maaten. "Submanifold sparse convolutional networks". In: arXiv preprint arXiv: 1706.01307 (2017).

[26] Théo Guénais et al. "BaCOUn: Bayesian Classifers with Out-of-Distribution Uncertainty". In: arXiv preprint arXiv: 2007.06096 (2020).

[27] Chuan Guo et al. "On Calibration of Modern Neural Networks". In: Proc. ICML 2017. JMLR.org, Aug. 2017, pp. 1321–1330.

[28] David Hall et al. "Probabilistic object detection: definition and evaluation". In: The IEEE Winter Conference on Applications of Computer Vision. 2020, pp. 1031–1040.

[29] 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.

[30] Kaiming He et al. "Deep residual learning for image recognition". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016,

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

[32] Dan Hendrycks and Kevin Gimpel. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: Proc. ICML 2017. JMLR.org, Oct. 2018. arXiv: 1610.02136.

[33] Dan Hendrycks, Mantas Mazeika and Thomas Dietterich. "Deep anomaly detection with outlier exposure". In: arXiv preprint arXiv: 1812.04606 (2018).

[34] Maximilian Henne, Adrian Schwaiger and Gereon Weiss. "Managing Uncertainty of AI-based Perception for Autonomous Systems." In: AISafety@ IJCAI. 2019.

[35] Maximilian Henne et al. "Benchmarking Uncertainty Estimation Methods for Deep Learning With Safety-Related Metrics". In: Proc.

SafeAI@AAAI 2020. Ed. by Huáscar Espinoza et al. Vol. 2560. CEUR Workshop Proceedings. 2020, pp. 83–90.

[36] José Miguel Hernández-Lobato and Ryan Adams. "Probabilistic backpropagation for scalable learning of Bayesian neural networks". In: International Conference on Machine Learning. 2015, pp. 1861–1869.

[37] Geoffrey Hinton, Oriol Vinyals and Jeff Dean. "Distilling the knowledge in a neural network". In: arXiv preprint arXiv: 1503.02531 (2015).

[38] Andrew G Howard et al. "MobileNets: Efficient convolutional neural networks for mobile vision applications". In: arXiv preprint arXiv: 1704.04861 (2017).

[39] Yen-Chang Hsu et al. "Generalized ODIN: detecting out-of-distribution image without learning from out-of-distribution data". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, pp. 10951–10960.

[40] Gao Huang et al. "Densely connected convolutional networks". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, pp. 4700–4708.

[41] Forrest N. Iandola et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 0.5 MB model size". In: arXiv preprint arXiv: 1602.07360 (2016).

[42] Alex Kendall and Yarin Gal. "What uncertainties do we need in Bayesian deep learning for computer vision?" In: Advances in Neural Information Processing Systems. 2017, pp. 5574–5584.

[43] Durk P. Kingma and Prafulla Dhariwal. "Glow: generative flow with invertible 1x1 convolutions". In: Advances in Neural Information Processing Systems. 2018, pp. 10215–10224.

[44] Anoop Korattikara Balan et al. "Bayesian dark knowledge". In: Advances in Neural Information Processing Systems 28 (2015), pp. 3438–3446.

[45] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". In: Advances in Neural Information Processing Systems. 2012, pp. 1097–1105.

[46] Jason Ku et al. "Joint 3D proposal generation and object detection from view aggregation". In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. 2018, pp. 1–8.

[47] Balaji Lakshminarayanan, Alexander Pritzel and Charles Blundell. "Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles". In: Advances in Neural Information Processing Systems 30. Curran Associates, Inc., 2017, pp. 6402–6413.

[48] Alex H Lang et al. "Pointpillars: fast encoders for object detection from point clouds". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019, pp. 12697–12705.

[49] Hei Law and Jia Deng. "CornerNet: detecting objects as paired keypoints". In: Proceedings of the European Conference on Computer Vision (ECCV). 2018, pp. 734–750.

[50] 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.

[51] Bo Li. "3D fully convolutional network for vehicle detection in point cloud". In: 2017 IEEE/RSJ International Conference on

Intelligent Robots and Systems (IROS). IEEE. 2017, pp. 1513–1518.

[52] Bo Li, Tianlei Zhang and Tian Xia. "Vehicle detection from 3D LIDAR using fully convolutional network". In: arXiv preprint arXiv: 1608.07916 (2016).

[53] Ming Liang et al. "Deep continuous fusion for multi-sensor 3D object detection". In: Proceedings of the European Conference on Computer Vision (ECCV). 2018, pp. 641–656.

[54] Shiyu Liang, Yixuan Li and Rayadurgam Srikant. "Enhancing the reliability of out-of-distribution image detection in neural networks". In: arXiv preprint arXiv: 1706.02690 (2017).

[55] Tsung-Yi Lin et al. "Feature pyramid networks for object detection". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, pp. 2117–2125.

[56] 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.

[57] Tsung-Yi Lin et al. "Microsoft COCO: common objects in context". In: European Conference on Computer Vision. Springer. 2014, pp. 740–755.

[58] Jakob Lindqvist et al. "A general framework for ensemble distribution distillation". In: arXiv preprint arXiv: 2002.11531 (2020).

[59] Shu Liu et al. "Path aggregation network for instance segmentation". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, pp. 8759–8768.

[60] Wei Liu et al. "SSD: single shot multibox detector". In: European Conference on Computer Vision. Springer. 2016, pp. 21–37.

[61] Andrey Malinin and Mark Gales. "Predictive uncertainty estimation via prior networks". In: Advances in Neural Information Processing Systems. 2018, pp. 7047–7058.

[62] Andrey Malinin, Bruno Mlodozeniec and Mark Gales. "Ensemble distribution distillation". In: arXiv preprint arXiv: 1905.00076 (2019).

[63] Damien Matti, Hazım Kemal Ekenel and Jean-Philippe Thiran. "Combining LiDAR space clustering and convolutional neural networks for pedestrian detection". In: 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). IEEE. 2017, pp. 1–6.

[64] Gregory P Meyer et al. "LaserNet: An efficient probabilistic 3D object detector for autonomous driving". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019, pp. 12677–12686.

[65] 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. 1–7.

[66] Dimity Miller et al. "Evaluating merging strategies for sampling-based uncertainty techniques in object detection". In: 2019 International Conference on Robotics and Automation (ICRA). IEEE. 2019, pp. 2348–2354.

[67] Eric Nalisnick et al. "Do deep generative models know what they don't know?" In: arXiv preprint arXiv: 1810.09136 (2018).

[68] Sang-Il Oh and Hang-Bong Kang. "Object detection and classification by decision-level fusion for intelligent vehicle systems". In: Sensors 17.1 (2017), p. 207.

[69] Aaron van den Oord, Nal Kalchbrenner and Koray Kavukcuoglu. "Pixel recurrent neural networks". In: arXiv preprint arXiv:

1601.06759 (2016).
[70] Yaniv Ovadia et al. "Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift". In: Advances in Neural Information Processing Systems. 2019, pp. 13991–14002.
[71] Jiangmiao Pang et al. "Libra R-CNN: towards balanced learning for object detection". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019, pp. 821–830.
[72] Janis Postels et al. "Sampling-free epistemic uncertainty estimation using approximated variance propagation". In: Proceedings of the IEEE International Conference on Computer Vision. 2019, pp. 2931–2940.
[73] Cristiano Premebida et al. "High-resolution LIDAR-based depth mapping using bilateral filter". In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC). IEEE. 2016, pp. 2469–2474.
[74] Charles R. Qi et al. "Frustum PointNets for 3D object detection from RGB-D data". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, pp. 918–927.
[75] Charles R. Qi et al. "PointNet: deep learning on point sets for 3D classification and segmentation". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, pp. 652–660.
[76] Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, pp. 7263–7271.
[77] Joseph Redmon and Ali Farhadi. "YOLOv3: an incremental improvement". In: arXiv preprint arXiv: 1804.02767 (2018).
[78] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016.
[79] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016, pp. 779–788.
[80] Shaoqing Ren et al. "Faster R-CNN: towards real-time object detection with region proposal networks". In: Advances in Neural Information Processing Systems. 2015, pp. 91–99.
[81] Hamid Rezatofighi et al. "Generalized intersection over union: a metric and a loss for bounding box regression". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019, pp. 658–666.
[82] Tim Salimans et al. "PixelCNN++: improving the PixelCNN with discretized logistic mixture likelihood and other modifications". In: arXiv preprint arXiv: 1701.05517 (2017).
[83] Adrian Schwaiger et al. "Is Uncertainty Quantification in Deep Learning Sufficient for Out-of-Distribution Detection?" en. In: Proc. AISafety@IJCAI2020. Vol. 2640. CEUR Workshop Proceedings. 2020, p. 8.
[84] Murat Sensoy, Lance Kaplan and Melih Kandemir. "Evidential Deep Learning to Quantify Classification Uncertainty". In: Advances in Neural Information Processing Systems 31. Curran Associates, Inc., 2018, pp. 3179–3189.
[85] Joan Serrà et al. "Input complexity and out-of-distribution detection with likelihood-based generative models". In: arXiv preprint arXiv: 1909.11480 (2019).
[86] Martin Simon et al. "Complex-YOLO: Real-time 3d object detection on point clouds". In: arXiv preprint arXiv: 1803.06199 (2018).

[87] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: arXiv preprint arXiv: 1409.1556 (2014).

[88] Jihoon Tack et al. "CSI: novelty detection via contrastive learning on distributionally shifted instances". In: Advances in Neural Information Processing Systems 33 (2020).

[89] Mingxing Tan and Quoc V. Le. "EfficientNet: rethinking model scaling for convolutional neural networks". In: arXiv preprint arXiv: 1905.11946 (2019).

[90] Mingxing Tan, Ruoming Pang and Quoc V. Le. "EfficientNet: scalable and efficient object detection". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, pp. 10781–10790.

[91] Aaron Van den Oord et al. "Conditional image generation with PixelCNN decoders". In: Advances in Neural Information Processing Systems. 2016, pp. 4790–4798.

[92] Chien-Yao Wang et al. "CSPNet: a new backbone that can enhance learning capability of CNN". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020, pp. 390–391.

[93] Yan Wang et al. "Pseudo-LIDAR from visual depth estimation: bridging the gap in 3D object detection for autonomous driving". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019, pp. 8445–8453.

[94] Oliver Willers et al. "Safety Concerns and Mitigation Approaches Regarding the Use of Deep Learning in Safety-Critical Perception Tasks". In: arXiv preprint arXiv: 2001.08001 (2020).

[95] Jim Winkens et al. "Contrastive Training for Improved Out-of-Distribution Detection". In: arXiv preprint arXiv: 2007.05566 (2020).

[96] Bichen Wu et al. "SqueezeSeg: convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LIDAR point cloud". In: 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE. 2018, pp. 1887–1893.

[97] Yan Yan, Yuxing Mao and Bo Li. "Second: sparsely embedded convolutional detection". In: Sensors 18.10 (2018), p. 3337.

[98] Bin Yang, Wenjie Luo and Raquel Urtasun. "Pixor: real-time 3D object detection from point clouds". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, pp. 7652–7660.

[99] Yurong You et al. "Pseudo-LIDAR++: accurate depth for 3D object detection in autonomous driving". In: arXiv preprint arXiv: 1906.06310 (2019).

[100] Jiahui Yu et al. "UnitBox: an advanced object detection network". In: Proceedings of the 24th ACM International Conference on Multimedia. 2016, pp. 516–520.

[101] Yin Zhou and Oncel Tuzel. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018.

[102] Yin Zhou and Oncel Tuzel. "VoxelNet: End-to-End Learning for Point Cloud-based 3D Object Detection". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, pp. 4490–4499.

[103] Martin Kunert. "Autonomous Driving Radar Requirements". Presentation. RADAR4FAD project. 2017.

[104] Krzysztof Czarnecki, Rick Salay. (2018) "Towards a Framework to Manage Perceptual Uncertainty for Safe Automated Driving". Presentation held at 1st International Workshop on Artificial Intelligence Safety Engineering (WAISE 2018). Västerås, Sweden, 2018.

9
Imprint

Fraunhofer Institute for Cognitive Systems IKS

Hansastrasse 32

80686 Munich, Germany

Authors

Maximilian Henne, maximilian.henne@iks.fraunhofer.de

Jens Gansloser, jens.gansloser@iks.fraunhofer.de

Adrian Schwaiger, adrian.schwaiger@iks.fraunhofer.de

Gereon Weiss, gereon.weiss@iks.fraunhofer.de

Munich 2021

10
Acknowledgment

This work was supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy through the Center for Analytics–Data–Applications (ADA-Center) within the framework of "BAYERN DIGITAL II".

Bavarian Ministry of Economic Affairs,
Regional Development and Energy

Contact

Adrian Schwaiger
Dependable Perception & Imaging
Tel. +49 89 547088-0
safe-intelligence@iks.fraunhofer.de

Fraunhofer IKS
Hansastrasse 32
80686 Munich, Germany

www.iks.fraunhofer.de