Uncertainty Estimation in One-Stage Object Detection

Florian Kraus[1] and Klaus Dietmayer[2]

***Abstract*— Environment perception is the task for intelligent vehicles on which all subsequent steps rely. A key part of perception is to safely detect other road users such as vehicles, pedestrians, and cyclists. With modern deep learning techniques huge progress was made over the last years in this field. However such deep learning based object detection models cannot predict how certain they are in their predictions, potentially hampering the performance of later steps such as tracking or sensor fusion. We present a viable approaches to estimate uncertainty in an one-stage object detector, while improving the detection performance of the baseline approach. The proposed model is evaluated on a large scale automotive pedestrian dataset. Experimental results show that the uncertainty outputted by our system is coupled with detection accuracy and the occlusion level of pedestrians.**

I. Introduction

Object detection is a crucial task for safe autonomous driving. With the introduction of deep learning the performance of object detection made a huge leap forward. However most of these methods have no measure of how certain they are in their output. When confronted with previously unseen data there is usually no way to measure if the model can deal with this input. For example a model trained on good weather data is faced with adverse weather situations.

Capturing uncertainty in object detection can provide a measure of how certain a model is in its output. Providing an uncertainty measure could also improve subsequent steps such as sensor fusion and tracking or be beneficial for active learning. Annotating images for object detection is a time consuming and costly process. To label only the data with the most information gain could help cutting time and costs.

Bayesian methods have a long history of providing an uncertainty measure. Bayesian Neural Networks (BNNs) are one way to apply Bayesian concepts into neural networks. In a BNN each weight is a random variables. As a result the output of such a model is also a random variable, providing a way to measure uncertainty.

There are two distinct concepts of uncertainty, *epistemic* and *aleatoric*. Epistemic uncertainty, is uncertainty which can be explained away given more data. This measures model uncertainty and is captured by Bayesian methods which become more confident the more data is used to train them. Aleatoric uncertainty, on the other hand, is data or problem inherent and cannot be reduced with more data. This includes sensor noise or ambiguities in the problem itself, e.g. matching problems in stereo depth map generation. It can be captured by explicitly modeling it as a model output.

[1]Florian Kraus is with Daimler AG, 89081 Ulm, Germany,

[2]Klaus Dietmayer is with Institute of Measurement, Control and Microtechnology, Ulm University, 89081 Ulm, Germany

Fig. 1 – Aleatoric variance for bounding box offset in y (vertical) direction. Variance is shown for prior box size of 55 by 20 pixels (height, width). Bright yellow indicates high uncertainty relative to the background. Notice the high variance for predictions which are slightly off vertically.

Until recently it was not practical to apply Bayesian methods to neural networks. Gal and Ghahramani [1] showed that a BNN can be approximated by incorporating dropout into an ordinary neural network. Kendall and Gal [2] also introduced aleatoric uncertainty measures predicted by the model itself. They applied both epistemic and aleatoric uncertainty estimation to instance and scene segmentation and depth estimation.

Recently their methods were incorporated into object detection by [3], [4]. Both used two-stage models which first generate proposals from a region proposal network and, in a second network, refine the box offsets and predict a class. Incorporating dropout into an one-stage object detector to capture uncertainty was first done by [5]. However, they did not use the methods provided by [1] to approximate the predictive distribution of the BNN, making their approach similar to model ensembles, which are an orthogonal method of capturing uncertainty.

In this article we rigorously incorporate the methods provided by [1] and [2] into an one-stage object detection framework. We provide all the necessary theory to do so and make our code publicly available[1]. Our model is a reimplementation of YOLOv3 [6] in TensorFlow [7] with an adjusted loss function to accommodate the Bayesian setting. We apply our method to train a pedestrian detector on an automotive pedestrian dataset and carry out experiments to compare the different uncertainty estimations. Furthermore we discuss the effects of using an one-stage object detector instead of a two-stage approach.

[1]`https://github.com/flkraus/bayesian-yolov3`

II. Related Work

Since [1] provided a straightforward way to incorporate Bayesian concepts into neural networks there has been numerous applications of these ideas. In the context of this article we are interested in applications to object detection or localization.

A. Uncertainty Estimation in Object Detection

Feng et al. [3] applied the methods proposed by [2] to a two-stage object detection network for vehicle detection in 3D Lidar data. They used the KITTI dataset [8] and a two stage Faster-RCNN architecture. Dropout was applied after the region proposal network (RPN) into the fully connected layers. They found that modeling aleatoric uncertainty improved the performance by 1 to 5 %, whereas the model with dropout uncertainty under-performed slightly to the baseline. As for the uncertainty, they found that detections which had a high Intersection over Union (IoU) with a ground truth object had lower epistemic uncertainty than those with lower IoU scores. For aleatoric uncertainty they found that the aleatoric uncertainty was correlated to the distance of objects.

Wirges et al. [4] also employed a two-stage approach but also added dropout to some of the convolutional layers. When applying dropout to all layers, they found that the model no longer converges. In contrast to [3], they not only trained on the car class but include pedestrians and cyclists. They found that cars have a lower epistemic uncertainty, which, they argued, could be due to the under representation of the pedestrian and cyclist class compared to the car class. Making the epistemic uncertainty a measure for underrepresented classes in a dataset.

Le et al. [9] incorporate aleatoric loss attenuation as proposed by [2] for the class prediction of an Single Shot MultiBox Detector (SSD) [10]. As another measure of uncertainty they used the fact that SSD spawns many boxes. They grouped all boxes with a certain overlap and used the grouped boxes to calculated an uncertainty measure, where the uncertainty is high if the grouped boxes disagree on their predictions. They used both methods to filter false positives.

Miller et al. [5] implemented a SSD architecture with dropout during test time. They employed a merging strategy of boxes from multiple forward passes based on the overlap. They achieved an increase in recall by 12.3% and 15.1% in precision. An uncertainty measure was calculated based on the merged boxes, effectively building a model ensemble, which are commonly used for uncertainty estimation.

Phan et al. [11] applied the framework of [2] to object localization. They calibrated the estimated uncertainties to give confidence intervals for the bounding box regression.

III. Method

To generate uncertainty estimations we incorporate Bayesian deep learning into the YOLOv3 one-stage object detection framework.

A. Bayesian Deep Learning

Bayesian Neural Networks (BNNs) are neural networks where each weight is a random variable instead of a fixed value. A prior distribution $p(\omega)$ is placed over the networks weights ω. Together with a likelihood function $p(y|x,\omega)$ a posterior distribution over the networks weights is obtained by applying Bayes' theorem

$$p(\omega|X,Y) = \frac{P(Y|X,\omega)p(\omega)}{P(Y|X)},$$

with the marginal likelihood

$$p(Y|X) = \int p(Y|X,\omega)p(\omega)d\omega,$$

and X, Y being the samples and ground truth of a given dataset. For regression tasks a Gaussian likelihood is used and for classification tasks a softmax likelihood.

For a new input x^* we have the predictive distribution

$$p(y^*|x^*,X,Y) = \int p(y^*|x^*,\omega)p(\omega|X,Y)d\omega.$$

In all but the simplest models the posterior and the predictive distribution are intractable.

One way to overcome this problem is to use variational inference where the posterior $p(\omega|X,Y)$ is approximated by a variational distribution $q_\theta(\omega)$ parametrised by θ. This distribution can be found by minimizing the Kullback–Leibler divergence $KL(q_\theta(\omega)||p(\omega|X,Y))$ or equivalently maximizing the *evidence lower bound* (ELBO):

$$\mathcal{L}_{VI}(\theta) := \int q_\theta(\omega)\log p(Y|X,\omega)d\omega - \mathrm{KL}(q_\theta(\omega)||p(\omega))d\omega \quad (1)$$

This approach results in an approximated variational predictive distribution

$$q_\theta(y^*|x^*) = \int p(y^*|x^*,\omega)q_\theta(\omega)d\omega. \quad (2)$$

In [1] they showed that using dropout in neural networks can be interpreted as variational inference. They interpreted dropout to perform on the weights rather than on the outputs of a layer. This means performing dropout can be seen as sampling weights from the variational distribution $q_\theta(\omega)$, effectively interpreting the trained weights of the network as the parameters θ of the variational distribution. Weights with no dropout layer are viewed as delta distributed with the value of the weight being the mean.

Gal showed that training a neural network with dropout and l2 weight decay is equivalent to maximizing the ELBO. The l2 weight decay loss is important since minimizing it corresponds with minimizing the KL term in (1). It is important to note that not only dropout can be seen as variational inference but all stochastic regularisation techniques which operate on the weights. The choice of regularisation technique manifests itself in an implicit prior distribution $p(\omega)$. For dropout minimizing the l2 weight decay loss is equivalent to minimizing the KL term between $q_\theta(\omega)$ and a normal distributed prior $p(\omega)$.

If the models likelihood function is a (multivariate) normal distribution $p(y^*|f^\omega(x^*)) = \mathcal{N}(y^*; f^\omega(x^*), \Sigma)$ Gal [12] showed that the expected value of the variational predictive distribution (2) can be approximated by Monte Carlo integration. This is done by sampling T sets of weights $\tilde{\omega}_t$ $(t = 1, \dots, T)$ from the variational (dropout) distribution:

$$\mathbf{E}_{q_\theta(y^*|x^*)}[y^*] \approx \frac{1}{T} \sum\nolimits_t f^{\tilde{\omega}_t}(x^*) \tag{3}$$

where $f^{\tilde{\omega}_t}(x^*)$ is a forward pass through the net using the sampled weights $\tilde{\omega}_t$. Essentially, performing T forward passes through the network f with dropout enabled. They called this process of summing up multiple forward passes with dropout enabled Monte Carlo (MC) dropout. The variance of the variational predictive distribution can be obtained by:

$$\begin{aligned}&\mathrm{Var}_{q_\theta(y^*|x^*)}[y^*] \\ &\quad\approx \Sigma + \frac{1}{T} \sum\nolimits_t f^{\tilde{\omega}_t}(x^*)^T f^{\tilde{\omega}_t}(x^*) \\ &\qquad - \left(\sum\nolimits_t f^{\tilde{\omega}_t}(x^*)\right)^T \left(\sum\nolimits_t f^{\tilde{\omega}_t}(x^*)\right)\end{aligned}$$

For classification tasks with either softmax (multiclass) or logistic (binary) likelihood $p(y^*|x^*, \omega) = s(f^\omega(x^*))$ the variational predictive distribution can be approximated by:

$$q_\theta(y^*|x^*) \approx \frac{1}{T} \sum\nolimits_t s(f^{\tilde{\omega}_t}(x^*)) \tag{4}$$

As an uncertainty measure for classification tasks Gal [12] proposed the mutual information (MI):

$$\begin{aligned}&\mathbf{I}[y, \omega|x, D_{\text{train}}] := \\ &\qquad \mathbf{H}[y|x, D_{\text{train}}] - \mathbf{E}_{p(\omega|D_{\text{train}})}\left[\mathbf{H}[y|x, \omega]\right]\end{aligned} \tag{5}$$

with $\mathbf{H}$ being the Shannon entropy. It can also be approximated by sampling multiple dropout forward passes. Consult [12] for more information. The mutual information is high for data where the model is uncertain on average but simultaneously, some weight configurations produced by the variational distribution yield outputs with high confidence.

B. One-Stage Object Detection with YOLOv3

YOLOv3 [6] is a one-stage object detector which predicts boxes at three different scales. The original input image is downsampled by a factor of 32, 16, and 8 respectively for the different output feature maps. For an input image of size 1024 by 1920 (height, width) this results in 32×60, 64×120, and 128×240 grids. Each cell of each grid encodes three different bounding boxes for three different prior box sizes (anchor boxes). Each bounding box consists of 4 bounding box offsets, 1 objectness score, and c class scores. The objectness score signals if there is an object present for a given prior and cell. Therefore, the three output grids are tensors of size $h_i \times w_i \times (3 \cdot (4 + 1 + c))$ with h_i and w_i being height and width of the i-th output grid $(i = 1, 2, 3)$. In total, YOLOv3 produces 120 960 boxes for an input image of size 1024 by 1920. The bounding boxes from the raw output are calculated as follows. Let c_x and c_y denote the distance of the upper left corner of a cell to the upper left corner of the input image and p_w, p_h denote the prior height and width of a given bounding box, then

$$\begin{aligned} b_x &= \sigma(y^*_x) + c_x, \\ b_y &= \sigma(y^*_y) + c_y, \\ b_w &= p_w \exp(y^*_w), \\ b_h &= p_h \exp(y^*_h). \end{aligned} \tag{6}$$

With the sigmoid (logistic) function $\sigma(x) := \frac{1}{1+\exp(-x)}$ and the model output $y^* = (y^*_x, y^*_y, y^*_w, y^*_h, y^*_{obj}, y^*_{c_1}, \dots, y^*_{c_n})$ consisting of $4 + 1 + c$ entries. The x and y coordinates of the bounding box center are denoted by b_x and b_y, width and height by b_w and b_h. The final objectness and class scores are calculated by applying the sigmoid activation function.

For calculating the loss each ground truth annotation (bounding box plus class label) is assigned to a single cell and prior (each cell holds 3 priors). Only the prior with the highest Intersection over Union (IoU) with the ground truth box receives a loss for the ground truth box (objectness, class and bounding box regression). After all ground truth boxes have been assigned, the remaining output cells and priors will receive only an objectness score loss to predict "no object".

In the following by prior we refer to the prior box sizes and not to the prior distribution.

C. Introducing Uncertainty to YOLOv3

We use a modified version of YOLOv3 which was implemented from scratch in TensorFlow[2]. The network architecture and ideas are the same as in the original implementation, however some changes were made. The loss was carefully revised to fit into the Bayesian setting. Another minor change regards the class scores. We replaced the sigmoid activation with a softmax activation. We made this change since in our experiments all classes are mutually exclusive, resulting in simpler code. To predict aleatoric uncertainty for the bounding box regression we add 4 more entries $(y^*_{\sigma_x}, y^*_{\sigma_y}, y^*_{\sigma_w}, y^*_{\sigma_h})$ to the output per bounding box.

To fit our model to the methods of Section III-A we need to model the output of our network as likelihood functions.

The bounding box regression is modelled as a multivariate normal distribution:

$$p(y^* | f^\omega_{loc}(x^*)) = \mathcal{N}(y^*; f^\omega_{loc}(x^*), \Sigma_{loc}),$$

with mean $f^\omega_{loc}(x^*)$ and diagonal covariance $\Sigma_{loc} = diag(\sigma^2_x, \sigma^2_y, \sigma^2_w, \sigma^2_h)$, where Σ_{loc} models the aleatoric uncertainty. Following [2], our aleatoric models include $(\sigma^2_x, \sigma^2_y, \sigma^2_w, \sigma^2_h)$ as an explicit output of the network. For the models without aleatoric uncertainty we set them to 1 during training and test time.

We use the negative log-likelihood loss for the localization:

$$\begin{aligned}&\mathcal{L}(y_{loc}, f^\omega_{loc}(x^*)) = -\log p(y_{loc}|f^\omega_{loc}(x^*)) = \\ &\quad \sum\nolimits_i \left(\frac{\sigma_i^{-2}}{2}\left(y_i - f^\omega_i(x^*)\right)^2 + \frac{1}{2}\log \sigma_i^2\right) + 2\log 2\pi\end{aligned}$$

with $i = x, y, w, h$.

[2] https://github.com/flkraus/bayesian-yolov3.

Note, that we model the raw outputs y^* to follow a multivariate normal distribution and not the final bounding box coordinates b_x, b_y, b_w, and b_h. This also means we have to invert eqs. (6) for our ground truth boxes to get to y_i[3].

Following [2] instead of predicting σ_i^2 we predict $s_i := \log \sigma_i^2$ to improve numerical stability:

$$\mathcal{L}(y_{loc}, f_{loc}^{\omega}(x^*)) = \sum_i \left(\frac{1}{2} \exp(-s_i) \left(y_i - f_i^{\omega}(x^*)\right)^2 + \frac{1}{2} s_i \right),$$

where we also ditched the constant term. To avoid NaN errors during training we also clip s_i to the interval $[-40, 40]$, this is reasonable since $\exp(\pm 40) \approx 1 \times 10^{\pm 17}$ covers a wide enough range. When setting $\sigma_i^2 = 1$ (or $s_i = 0$) this loss is equivalent to the standard l2 regression loss $\frac{1}{2} \left\| y_{loc} - f_{loc}^{\omega}(x^*) \right\|^2$. We call those predicted variances σ_i^2 aleatoric variances or uncertainty.

The objectness score is modeled as a logistic likelihood:

$$p(y^* = c \,|\, f_{obj}^{\omega}(x^*)) = c \cdot \sigma(f_{obj}^{\omega}(x^*)) + (1 - c)(1 - \sigma(f_{obj}^{\omega}(x^*)))$$

with $c \in \{0, 1\}$, representing the presence or absence of an object. For the loss we use the negative log-likelihood, which, for the sigmoid likelihood, is the same as the cross entropy loss:

$$\mathcal{L}(y_{obj}, f_{obj}^{\omega}(x^*)) = - y_{obj} \cdot \log \sigma(f_{obj}^{\omega}(x^*)) - (1 - y_{obj}) \cdot \log(1 - \sigma(f_{obj}^{\omega}(x^*))),$$

where $y_{obj} \in \{0, 1\}$ is the ground truth label.

The classification part is modeled as a softmax likelihood:

$$p(y^* | f_{cls}^{\omega}(x^*)) = \text{softmax}(f_{cls}^{\omega}(x^*)),$$

again with the standard cross entropy loss (negative log-likelihood). Modelling all our losses as negative log likelihood allows us to use the techniques proposed by [1].

We combine all three losses into a single likelihood function by taking the product of the three individual likelihood functions:

$$p(y^* | f^{\omega}(x^*)) = p(y_{loc}^* | f_{loc}^{\omega}(x^*)) \cdot p(y_{obj}^* | f_{obj}^{\omega}(x^*)) \cdot p(y_{cls}^* | f_{cls}^{\omega}(x^*))$$

The combined loss is the sum of the three individual losses which is the same as the negative log-likelihood of the combined likelihood function

$$\mathcal{L}(y, f^{\omega}(x^*)) = \mathcal{L}(y_{loc}, f_{loc}^{\omega}(x^*)) + \mathcal{L}(y_{obj}, f_{obj}^{\omega}(x^*)) + \mathcal{L}(y_{cls}, f_{cls}^{\omega}(x^*)).$$

We also add a l2 weight decay $\mathcal{L}_{l2}(\omega) = \frac{1}{2} \lambda \left\| w \right\|^2$ to the loss. In doing so minimizing our loss becomes equivalent to maximizing the ELBO (1).

[3]The inverting of the bounding box formulas for the ground truth is also mentioned in the original YOLOv3 paper but is not present in the code. Instead the loss is $\sigma(y_x^*) - y_x$. Also, their use of the l1 loss instead of an l2 loss does not allow us to view this as the negative log-likelihood of a normal distribution.

D. Bayesian YOLO

We use the same network architecture as YOLOv3 which uses darknet53 [6] as a feature extractor (base net) and on top of that several convolutional layers mixed with upsampling layers to achieve the different downsampling scales of 32, 16, and 8. All layers are convolutional layers making it invariant to the input size, only the prior sizes must be adjusted accordingly.

To predict epistemic uncertainty we add dropout to our model. Before each output feature map we add dropout to five convolutional layers. In total 15 dropout layers are added. The dropout is applied right after the convolutional layer and before the batch normalization layer. Note that we did not remove the batch normalization layer. Batch normalization can be seen as another non linear activation and does not interfere with the variational inference as long as we place the dropout layer right after the convolutional layer. We used a dropout rate of 0.1. We also experimented with a dropout rate of 0.2 but this yielded worse model performance.

Leaving the batch normalization layers stabilizes the training. Another advantage is that we can easily use weights of a pre-trained model without dropout to jump-start the training of the model with dropout.

Our model predicts boxes at three different scales and for nine different prior sizes. This can be interpreted as a combined model consisting of nine different (sub-)models which share most weights. Each of these sub-models predicts n_i boxes ($i \in \{1, \ldots, 9\}$), with n_i depending on the input image size. Each of these boxes must be interpreted as the output of the likelihood functions introduced in Section III-C. For each box the regression parameters are calculated by approximating the expected value of the variational predictive distribution by applying (3) and the class and objectness scores by approximating the predictive distribution via (4). For our model with aleatoric loss and MC dropout enabled the aleatoric variance is calculated by averaging out multiple forward passes. For this process to make sense we utilize the fact that the box predictions of YOLOv3 are locally restricted to the cell they are predicted by. Therefore multiple forward passes of the same box with dropout enabled still refer to the same location in the input image.

This is different to SSD where the bounding box offsets are not restricted to the enclosing cell but can, in theory, shift a box over the whole image. This causes problems if multiple dropout forward passes of one output predict offsets for different objects. This might be the reason why [5] grouped boxes of multiple forward passes based on their overlap rather than averaging out the network output to approximate the variational predictive distribution.

The nine in one model of YOLOv3 is quite different to the two-stage approach taken by [3]. The last fully connected layers which they used to predict the uncertainties is the same for all region proposals. The effect is that the uncertainty estimations come all from the same model with the same weights which is not the case for our Bayesian YOLO.

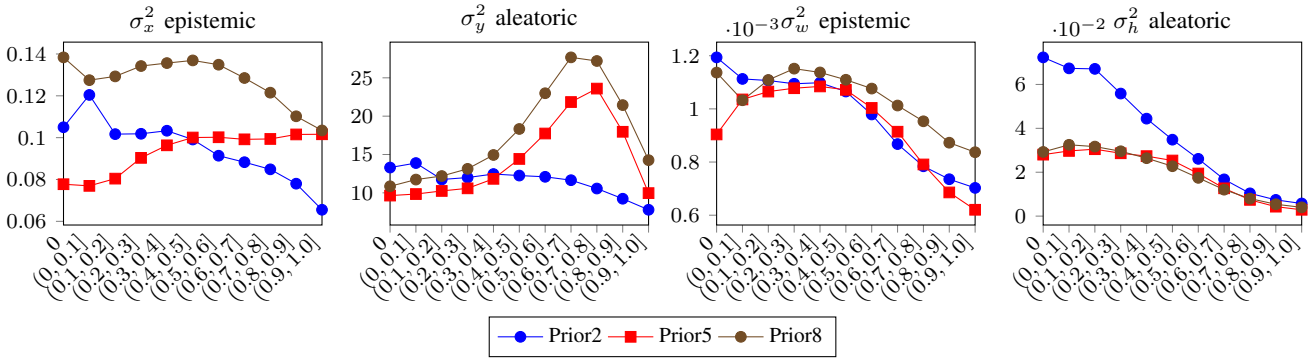


Fig. 2 – Uncertainties compared to intersection over union (IoU), we show one prior size from each output scale. The x-axis shows the IoU ranges.

E. Training and Dataset

All models are first trained without dropout and aleatoric loss for 125 000 iterations. Each iteration is the forward and backward pass of one batch. We used a batch size of 16 for all models. We then use the checkpoint after 125 000 iterations to train three models. One model with dropout enabled (as described above), one without dropout but the aleatoric loss enabled and one combined model with dropout and aleatoric loss enabled (referred to as aleatoric+epistemic). The initial training without dropout and aleatoric loss is crucial for stable training as found by [4] and confirmed by us.

We continued the training for the model without any uncertainty as a baseline. All models were trained for further 375 000 iterations to a total of 500 000 with the first 125 000 iterations shared between all four models. For all experiments we used an Adam optimizer [13] with an initial learning rate of 0.001. The prior sizes are determined by applying the k-means clustering to the box sizes of the training dataset as described in [14]. Training one model for 500 000 iterations took about six to seven days on a Nvidia Tesla V100.

For our experiments we use the EuroCity Persons (ECP) dataset [15] which consists of roughly 47000 images of which around 7000 are captured at night. In total around 218000 pedestrians and 20000 riders are labeled[4]. The dataset provides annotations for the level of occlusion of every object. The labels have the following granularity: no occlusion, $11\% - 40\%$, $41\% - 80\%$, and $> 80\%$. We use these occlusion labels in our experiments.

T	LAMR	time (ms)
no dropout	17.75	**120**
5	17.97	203
10	17.58	257
20	17.37	411
40	17.33	743
50	17.34	912
100	**17.28**	1907

TABLE I – Performance difference for different number of stochastic forward passes and inference time for full image size of 1024×1920.

		LAMR		
Training Data	Model	Day	Night	Steps
day	baseline	18.97	29.96	300 000
	aleatoric	**18.20**	**28.04**	350 000
	epistemic	18.62	29.26	300 000
	aleatoric + epistemic	18.89	28.37	400 000
night	baseline	32.42	31.25	325 000
	aleatoric	32.03	28.17	400 000
	epistemic	29.57	26.56	475 000
	aleatoric + epistemic	**29.71**	**26.11**	500 000
day + night	baseline	17.82	24.69	300 000
	aleatoric	17.79	**23.27**	425 000
	epistemic	17.75	25.14	425 000
	aleatoric + epistemic	**17.37**	24.47	425 000

TABLE II – Model Performance on ECP test set, measured with LAMR metric (lower is better).

IV. Experiments and Results

To train our models we split the training data into day, night and day + night. Each split was used to train 4 models as described in Section III-E. The models were trained to predict pedestrians and riders. We applied non-maximum suppression for each class individually.

To measure model performance we use the evaluation scripts provided by [15]. The performance metric used is the *log average miss-rate* (LAMR), lower scores in this metric are better. We only evaluate for pedestrians using the *reasonable* difficulty and *ignore* setting of the evaluation script such that riders detected as pedestrians are ignored. These are the same settings used in [15]. First, we selected the best performing training iteration on the validation set. For each model we evaluated checkpoints every 25 000 iterations starting at 300 000 up to 500 000. For the models trained on the day or day + night split we selected the model performing best on the day validation set. The models trained only on night data were evaluated on the night validation set. Then, we evaluated the best performing models on the day and night test splits. To evaluate the epistemic models we used MC dropout, with $T = 50$ forward passes per image.

[4]The training and validation splits including annotations are available to the public at `https://eurocity-dataset.tudelft.nl`.

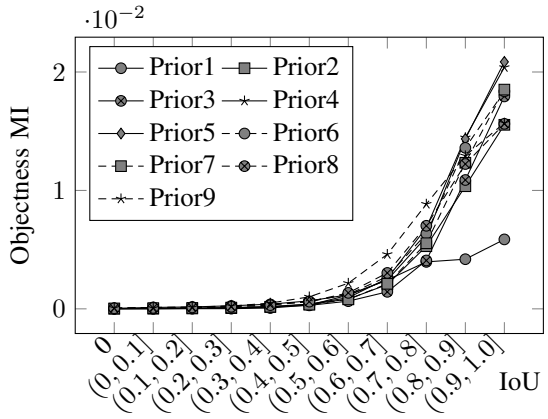


Fig. 3 – Mutual information of the objectness score plotted against the overlap with an ground truth object.

(a) Prediction for fully visible pedestrian.

(b) Prediction for heavily occluded pedestrian.

Fig. 4 – Comparing aleatoric uncertainties for occluded and fully visible pedestrians. Both Predictions were produced by prior 3. Note the drastically increased variance in width and height for the heavily occluded pedestrian.

A. Performance Evaluation

In Table II the performance of the different models are compared. We found that employing the aleatoric loss and dropout slightly increased model performance. When training on night data we see a huge performance gain from 31.25 to 26.11 evaluating on the night dataset. Both models with dropout outperformed the baseline model by a large margin. This might be due to overfitting of the baseline approach. The night training set only contains 4200 images compared to 24000 of the day training set. Using dropout and aleatoric loss might have prevented the models from overfitting.

In Table I we evaluate the effect of different numbers of forward passes for the MC dropout. We use our best performing model *aleatoric + epistemic* trained on day and night data. The model is evaluated on the day test set. Performance and inference time are presented for different T and for the standard dropout approximation with only one forward pass. As expected the number of forward passes increases model performance. An interesting fact from this evaluation is, that the standard dropout procedure did not perform significantly worse, outperforming the model with $T = 5$ MC dropout forward passes.

To save time the input images are only passed once through the base feature extractor, since no dropout is applied during this stage of the network. The output of the feature extractor is stacked T times to a single batch with T identical entries and then processed in parallel by the subsequent layers. Although done in parallel this still drastically impacts performance. However, for a full sized image of 1024 by 1920 going from one forward pass to ten doubles the inference time from 120ms to 257ms, which, with further optimizations applied, could still be manageable.

B. Uncertainty Estimations

When evaluating the uncertainty measures we find that the uncertainties between the individual priors are not calibrated, this can be seen in Fig. 2, where the average uncertainty for prior 2 clearly differs from prior 8. As a result we can only compare the uncertainties for each prior individually. This is different from the uncertainty estimation of [3], where they used a two-stage object detector. The two-stage detector has a single output to predict the class and box refinements of the region proposal network. Making the uncertainty estimation comparable between different boxes.

Since both YOLOv3 and SSD predict boxes on multiple feature maps, using uncertainty estimations for these type of networks requires additional work to calibrate the outputs. Investigation of suitable calibration methods will be reserved for future research.

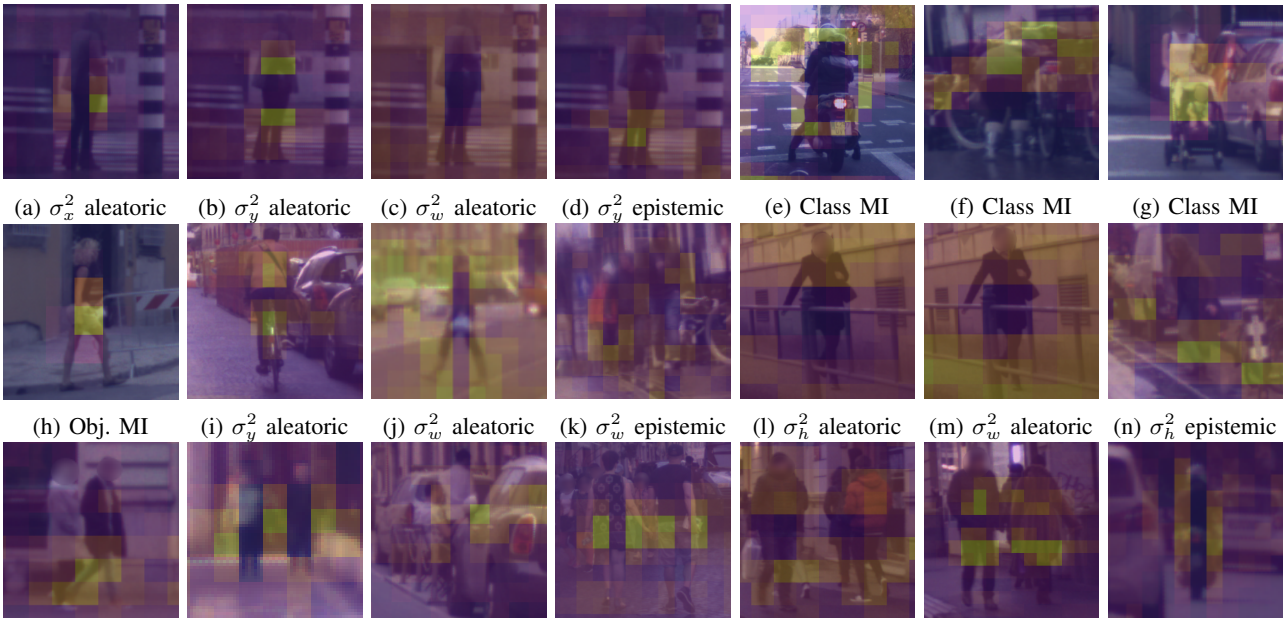
When comparing the variance for the regression task we found for both the aleatoric and the epistemic uncertainty that the variance for the bounding box offset was higher than for the width and height adjustment. This seems plausible, since the prior sizes were carefully chosen to fit the training data well, so less adjustment is needed.

We carried out a qualitative analysis of the uncertainty estimates. We overlaid images with the uncertainty estimations for each cell of each prior, see Fig. 5. For the regression uncertainties, both aleatoric and epistemic, it is clearly visible that the center cell of each object has the lowest uncertainty. The cells around them have higher uncertainty which decreases further away, with large parts of the background having low uncertainty (see Fig. 1).

The variance for the x and y offset predictions are particularly interesting. For the x variance we found multiple cases where the uncertainty is low in a center strip and high to the left and right. Conversely for the y variance we find the uncertainty to be high above and below the center (see Figs. 5a and 5b). This phenomenon happens for both epistemic and aleatoric uncertainty estimations. Although it seems that it is more pronounced for the aleatoric uncertainty.

The uncertainty estimations for the height and width do not show such clear patterns, although we still find the center to have the lowest uncertainty. Altogether the epistemic variance seems more spread out compared to the aleatoric.

Having low uncertainty estimations for the cell which is responsible for predicting an object is what we would expect from a sensible uncertainty estimations. We want to recall, that in the YOLO framework only one cell is responsible for predicting an object. This differs from the SSD framework where multiple cells are trained to predict a single ground truth object. On the other hand it is import to have higher



(a) σ_x^2 aleatoric (b) σ_y^2 aleatoric (c) σ_w^2 aleatoric (d) σ_y^2 epistemic (e) Class MI (f) Class MI (g) Class MI

(h) Obj. MI (i) σ_y^2 aleatoric (j) σ_w^2 aleatoric (k) σ_w^2 epistemic (l) σ_h^2 aleatoric (m) σ_w^2 aleatoric (n) σ_h^2 epistemic

(o) σ_y^2 epistemic (p) σ_x^2 epistemic (q) σ_x^2 epistemic (r) σ_x^2 aleatoric (s) σ_h^2 epistemic (t) σ_y^2 aleatoric (u) σ_x^2 aleatoric

Fig. 5 – Uncertainty estimations visualized. Bright yellow indicates high uncertainty compared to the rest of the image. Note the low uncertainty at the center of each object for the regression variances.

uncertainty for cells which are slightly off, indicating that the bounding box is not quite right. And we do not care as much about the uncertainty estimations for the background.

The effect of having low uncertainties in the center can also be seen in the quantitative analysis carried out in Fig. 2. We plotted the different uncertainty measures against the overlap (IoU) with a ground truth object. We clearly see the uncertainty going down for an overlap greater than 0.7. The aleatoric variance for the y offset even ramps up for detections which are slightly off with an overlap of around 0.8 and then going down for more precise detections.

We also evaluated the Mutual Information (MI) (see eq. (5)) for the class and objectness score. The MI for the class score is mostly low but sometimes shows higher values around objects (see Fig. 5e). For some difficult objects, such as strollers, or pedestrians standing in front of a bicycle the MI goes up (see Figs. 5f and 5g). Since the model was trained on riders and pedestrians a person standing in front of a bicycle has both features for the pedestrian and the rider class increasing the MI for the class prediction.

The MI for the objectness score does not predict sensible uncertainty information when evaluating it. The uncertainty is very low for the complete background and only high on objects. With the center of the object having the highest uncertainty (see Fig. 5h). This can also be seen in Fig. 3 where the MI correlates positively with the IoU with an ground truth object.

This behaviour is likely a result of the training process. The objectness score is the only output which is also trained on background. Since most parts of an image are background this causes the model to become very confident on the background. Unfortunately this makes the MI of the objectness not a practical measure of uncertainty, since correct predictions have high and incorrect ones low uncertainty. Maybe this effect could be counteracted by training for more iterations or trying to balance out the background and foreground examples during training as SSD does. The other uncertainty estimations do not suffer from such effects since the other model outputs only incur a loss for positive examples.

C. Occlusion

We evaluated how the occlusion of an ground truth object affects the uncertainty estimations. We found the aleatoric uncertainty for the bounding box regression to be positively correlated with the occlusion of the detected object.

To evaluate this we selected for each ground truth object the model output which is responsible for predicting it (similar to the training process). We calculate the Pearson correlation coefficient between the occlusion level of the matched ground truth (0%, $11\% - 40\%$, $41\% - 80\%$, $> 80\%$). We find that the variance of the bounding box regression parameters for width and height have a strong positive correlation between the occlusion and the predicted variance, see Table III. This seems plausible since occluded objects in the ECP dataset have an estimated bounding box of the full extent. These, by humans estimated, boxes are used during the training phase forcing the model to guess the correct width and height of the object, possibly predicting high aleatoric variance to keep the loss low. We also found a slight correlation between the occlusion and aleatoric variance for the bounding box offsets. The epistemic uncertainty did not show any correlation with the occlusion.

A qualitative analysis of this can be found in Fig. 4 where we compared the aleatoric uncertainty of an occluded and a

		Prior1	Prior2	Prior3	Prior4	Prior5	Prior6	Prior7	Prior8	Prior9
σ_w^2	r	0.070	0.359	0.490	0.499	0.474	0.466	0.392	0.254	0.025
	p	0.374	< 0.000	< 0.000	< 0.000	< 0.000	< 0.000	< 0.000	< 0.000	0.118
σ_h^2	r	-0.005	0.214	0.383	0.202	0.239	0.349	0.445	0.371	0.140
	p	0.947	< 0.000	< 0.000	< 0.000	< 0.000	< 0.000	< 0.000	< 0.000	< 0.000
data count		164	408	828	1821	2668	3252	5635	5825	3936

TABLE III – Pearson correlation coefficient r between occlusion and aleatoric uncertainty for w and h. The last row shows the number of detections used to calculate r for each prior size. This was calculated using the epistemic + aleatoric model trained on day and night data. The occlusion was evaluated on the day validation set. We found the same tendencies for different models and when evaluating on the night validation set.

fully visible pedestrian. Both boxes are produced by prior 3 so the uncertainty measures are comparable. We found that when comparing the uncertainties of different priors an occluded box can have lower uncertainty than a non-occluded box from a different prior, which again, stresses the importance of keeping the uncertainty measures of different priors separate or calibrate them, to make them comparable.

V. Conclusion

In this work we showed how to incorporate the methods proposed by [1], [2], and [12] into a modern one-stage object detection framework. We showed how to approximate the variational predictive distribution of our model and how to capture the different uncertainty measures. We provided the code to train models with epistemic and aleatoric uncertainties in TensorFlow. Along with a re-implementation of YOLOv3 in TensorFlow which supports training from scratch.

We showed in a quantitative and qualitative analysis that the uncertainty measure correlates with the overlap with objects. Predictions which are slightly off have higher uncertainty than more precise ones, which is exactly what we would expect from a sensible uncertainty estimation. This also means the uncertainty estimation for the bounding box regression could be used to improve non-maximum suppression, by favouring boxes with lower bounding box regression variance. We also found the aleatoric uncertainty to be correlated with the occlusion of objects making it a measure of problem inherent ambiguities.

In future work we plan to calibrate the uncertainty estimation for different prior sizes to make them comparable to each other. A viable solution0 seems to be to incorporate the techniques proposed by [11] to calibrate the bounding box regression uncertainty. Also we want to address the effect of adverse weather situations such as rain or fog on the uncertainty measure. This work provides us with the necessary framework and theory to further research uncertainty estimations in object detection tasks.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union under the H2020 ECSEL Programme as part of the DENSE project, contract number 692449.

References

[1] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with Bernoulli approximate variational inference," in *4th International Conference on Learning Representations (ICLR) workshop track*, 2016.

[2] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5574–5584.

[3] D. Feng, L. Rosenbaum, and K. Dietmayer, "Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov 2018, pp. 3266–3273.

[4] S. Wirges, M. Reith- Braun, M. Lauer, and C. Stiller, "Capturing object detection uncertainty in multi-layer grid maps," *arXiv:1901.11284 [cs.RO]*, Jan 2019.

[5] D. Miller, L. Nicholson, F. Dayoub, and N. Sünderhauf, "Dropout sampling for robust object detection in open-set conditions," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1–7.

[6] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv:1804.02767 [cs.CV]*, Apr 2018. [Online]. Available: https://pjreddie.com/darknet/yolo/

[7] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: www.tensorflow.org

[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[9] M. T. Le, F. Diehl, T. Brunner, and A. Knol, "Uncertainty estimation for deep neural object detectors in safety-critical applications," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov 2018, pp. 3873–3878.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37.

[11] B. Phan, R. Salay, K. Czarnecki, V. Abdelzad, T. Denouden, and S. Vernekar, "Calibrating uncertainties in object localization task," *arXiv:1811.11210 [cs.LG]*, Nov 2018.

[12] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, University of Cambridge, 2016.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[14] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[15] M. Braun, S. Krebs, F. Flohr, and D. Gavrila, "Eurocity persons: A novel benchmark for person detection in traffic scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.