Hybrid bayesian convolutional neural network object detection architectures for tracking small markers in automotive crashtest videos

Felix Neubürger*[1], Daniel Gierse[1], and Thomas Kopinski[1]

[1]University of Applied Science South Westphalia

Abstract

Automotive crash tests are an important aspect of everyday safety, where accurate measurements and evaluations play a crucial role. In order to automate this process, we implement a bayesian hybrid computer vision model to detect two different kinds of target markers. These are commonly used in crash tests, by e.g. automotive manufacturers, to aid the localisation of the car's positional data by attaching these markers at different positions to the vehicle. A tracking algorithm is subsequently used to add contextual time information to the marker objects. The extracted information can then be used in downstream tasks to calculate important metrics for the crash test evaluation, e.g. the speed, momentum, acceleration and trajectory at particular parts of the car during different stages of the crash test. The model consists of a pre-trained Faster-RCNN for the region proposals with the addition of a bayesian convolutional neural network to estimate a statistical uncertainty on the model's classifications. This uncertainty estimation can be used as a tool to improve safety in uncertain edge cases in videos where lighting conditions and light reflections are not optimal. Our pipeline achieves an average recall and precision of 0.89 and 0.99, respectively, when applied to test data. This outperforms the recall of state of the art models like the Faster-RCNN Resnet-152 by more than 28% while delivering slightly better precision, increasing robustness in most of the tested use-cases.

*Corresponding Author: neubuerger.felix@fh-swf.de

1 Introduction

When detecting and classifying objects in images and videos the inherent uncertainty of what exactly the detected object is, is often not quantifiable. In this work, conducted in cooperation with the BMW AG [1] we develop an approach to quantify the model's predictive uncertainty of an object detection and classification pipeline by using a bayesian convolutional neural net on top of different object detectors. We apply this method on high resolution automobile crash test videos to detect, identify and track the target markers attached to the car. These target markers are used by engineers to extract important physical and mechanical quantities in their crash test evaluations. Among other things, these quantities consist of the speed and trajectory of different parts of the car, allowing to derive further information from the collected data and helping in modelling the deformation process of the car on, and after, the monitored impact. In highly dynamic environments with varying lighting conditions and reflections an object detector must be robust and yield certain results. If additionally an uncertainty for a prediction can be estimated this adds another layer of trust in the machine learning algorithm. The developed approach helps to very accurately track the target markers with a robust prediction of the marker type to avoid mix-ups in this highly dynamic environment. In addition to the robustness and accuracy the model's uncertainty on a prediction is measured. The developed approach to object detection and classification

[1]This research was conducted at the Data Science Lab at of the University of South Westphalia in close collaboration with the engineers from the Requirements and Strategy Vehicle Safety department at BMW (EG-30).

https://doi.org/10.7557/18.6802

is easily applicable to other computer vision tasks where aleatoric uncertainty is important in decision making. The structure of this contribution is laid out as follows: First, we show a brief overview of existing methods in object tracking and bayesian image classification and then contrast them with our approach in 2. Subsequently we describe the used dataset. Finally we present the model architectures and evaluation results of our analysis in 4 and discuss the results and implications in 5.

2 Related Work

Previous works have given examples on how to combat the inherent lack of an uncertainty estimation in neural network models. Especially in the case of computer vision the task of calculating such an uncertainty on the object's bounding box itself requires a redesign of the current state of the art neural network architectures. In [6] the authors use a modified non maximum suppression algorithm to acquire such an uncertainty measurement on proposed object regions. Other models like the Faster-RCNN require more modifications to the model's architecture. In [5] the authors benchmark a modified Faster-RCNN on the MNIST dataset and apply it to a dataset from molecular biology to enhance the classification output. In this work we combine methods of Faster-RCNN and Bayesian-CNN networks to efficiently detect small objects in videos while also quantifying a statistical uncertainty on classification predictions where a standard neural network yields a confidence not deemed trustworthy enough.

3 Data and model structure

The data for the training process was sampled from different videos provided by BMW as well as using public clips from NCAP crash tests on YouTube. As the latter were only available as compilations, these videos had to be cut to only show one particular test scenario per input video. Images taken from these clips were then normalized and labeled, the exact distribution of annotated samples can be found in tab. 1. The resolutions of the input videos range from 1920x1080 pixels to 2560x2200 pixels, where the largest portion falls into the first

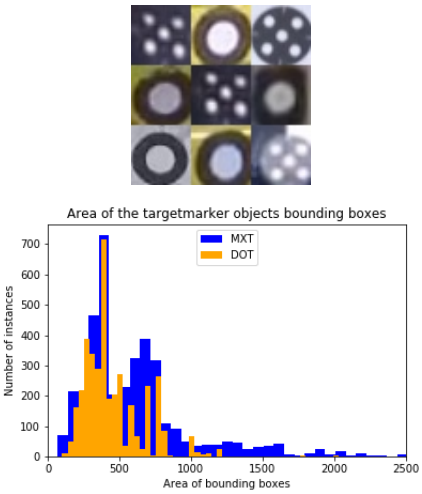

Figure 1: Top: MXT (5 white circles) and DOT crops as detected by the Faster-RCNN. These are used as input for the Bayesian Neural Network. Bottom: Distribution of the areas (in pixels) of the two aforementioned marker types.

category. Resizing and data augmentation during training and inference time was handled directly by a Faster-RCNN ResNet-152, taken from the tensorflow model zoo [7] [2] where all models are pre-trained on the COCO 2017 dataset. Additionally it was also tasked with the detection of the two different types of target markers displayed in figure 1. The training data for the Bayesian

		Faster RCNN		BNN/LeNet	
task	videos	MXT	DOT	MXT	DOT
train	75	3181	2741	6432	6432
test	19	333	148	1440	1440

Table 1: Distribution of labeled training and test samples taken from the crash test videos for the different networks.

Neural Network was generated by utilizing the previously trained object detector to automatically create samples of target regions from the video clips. Afterwards these were manually cleaned up from misclassifications and misdetections. Some examples of the target objects can be found on the left in fig. 1. The histogram on the right side of the same figure clearly shows that the area of the vast majority of possible target regions is below

32^2 square pixels. There are some outliers in the object sizes due to the various camera positions leading to a wide range of distances from camera to object, these positions are presented in fig. 2). From this distribution we set a unified input size of only 16x16 pixels for the Bayesian Neural Network, with a varying number of convolutional layers as can be seen in tab. 3. One of the key aspects of Bayesian Neural Networks is their ability to model aleatoric and epistemic uncertainty by utilizing probability distributions instead of fixed values, leading in our case of Gaussian priors, to a doubling in parameters (μ, σ instead of just the weight w). As the epistemic uncertainty is not quantifiable in our application due to the range of different lighting conditions, camera angles and background colors, we focus on the aleatoric uncertainty. This is achieved by using a technique called *Variational Inference* which focuses on the approximation of the true posterior densities, avoiding the problem of intractable calculations for the latter, and presents a more scalable alternative to Monte Carlo Markov Chains. In order to achieve this it utilises a family of distributions with variable parameters to optimize over and among them tries to find the best approximation $q(z)$ of the true posterior distribution $p(z|x)$. The likelihood is determined by using a metric called the *evidence lower bound*(ELBO), a slightly modified version of the negative Kullback-Leibler (KL) divergence without the oftentimes intractable computation of the evidence log $p(x)$:

$$KL(q(z)||p(z|x)) = \mathbb{E}[\log q(z)] - \mathbb{E}[\log\ p(z|x)]$$
$$KL(q(z)||p(z|x)) = \mathbb{E}[\log q(z)] - \mathbb{E}[\log p(z,x)] + \log p(x)$$
$$ELBO(q) = \mathbb{E}[log\ p(z,x)] - \mathbb{E}[\log q(z)]$$

Maximising the ELBO in turn leads to a minimisation of the KL divergence and thus a better approximation of the target distribution.[3]As an exact calculation of the former is again often times infeasible, it was proposed by [4] to approximate this function by Monte Carlo sampling from the variational posterior q, where the samples are gathered by using a modified version of the reparameterisation trick from [8]. This method relies on taking a sample $\epsilon \sim \mathcal{N}(0,1)$ from a distribution of parameter-free noise and then shifting and scaling it by the parameters $\theta = (\mu,\rho)$ of q, where $\sigma = \log(1+\exp(\rho))$. The transformation generates a sample according to the deterministic function $t(\theta,\epsilon) = \mu + \log(1+\exp(\rho) \circ \epsilon$ which can then be used to optimize the cost function for the learning process.

Figure 2: Examples of various camera perspectives and distances found in the video material used in training and testing.

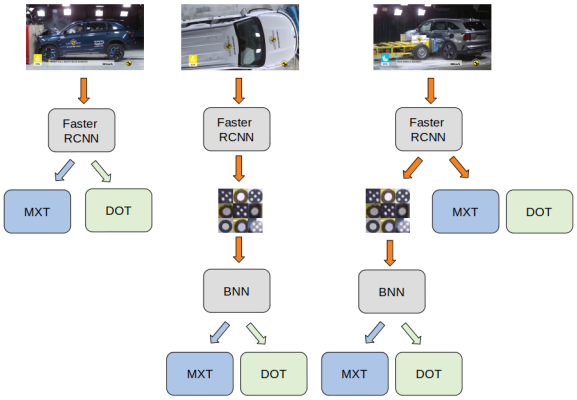

Figure 3: Processing pipelines for testing. Left: First implementation based on Faster-RCNN for target detection. Middle: Faster-RCNN delivering ROIs which are then classified by a BNN using multiple forward passes to determine the final result + confidence. Right: Third iteration, relying on the Faster-RCNN and only falling back onto the BNN once the detection score is below a certain threshold.

To mitigate the impact of the small target regions on model and pipeline performance, an additional dataset was created by resizing the

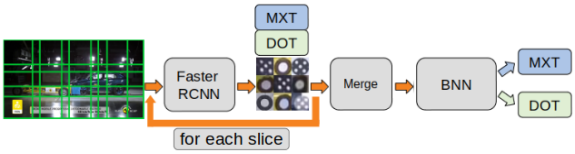

Figure 4: The fourth iteration of the pipeline segments the image into 15 different slices evaluated by the Faster-RCNN. The detected regions of the slices are merged by removing duplicates, results are furthermore evaluated by the BNN and only accepted when both agree upon the result.

original images to 1920x1080 pixels. Each of these were furthermore split into 15 partly overlapping slices of 500x440 pixels, allowing the underlying Faster-RCNN ResNet-152 to work on the input without losing any more of the already sparse information. The resulting overlap of at least 80 pixels in each direction in between these crops ensures that for each marker there is at least one slice where its bounding box will be able to be completely captured.

These trained networks are then used as parts of different processing pipelines. The ones that can be seen in fig. 3 use different combinations of the same underlying networks to achieve their goals of correctly detecting, classifying and tracking each target marker present in the recordings of the crash tests, yet they are all based upon using the unsliced input. The pipeline in fig. 4 however operates on a Faster-RCNN trained on image slices while reusing the same BNN. The latter was chosen due to the results displayed in table 3, which were gathered by classifying various test images, leading to the decision to use BNN3 for all pipelines that involve a Bayesian Neural Network, namely pipelines 2, 3 and 4. The subsequent tracking process of each individual marker is handled by object trackers using the CSRT algorithm [9]. This offers high accuracy and robust tracking of objects, even when there are rapid changes in trajectory. One noticeable drawback of this algorithm is its unstable predictions once the target gets occluded for several consecutive frames, coming into effect when there is deformation of the car during the crash. This unwanted behavior gets filtered out by measuring the overlap of the predicted bounding boxes of each

Figure 5: Sample output from the developed pipeline. This image shows the small size of the target regions. The different marker types are found accurately in the high resolution image.

active tracker with the ones provided by the Faster-RCNN, using *Intersection over Union* (IoU) as a metric, which is defined as

$$\mathrm{IoU}(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}, \quad (1)$$

with the bounding boxes A and B. Every match leads to a renewed confirmation of the object, which in turn keeps it alive longer. Once a target has not been confirmed for a certain amount of consecutive frames, its status is set to inactive and the positional data up to the last successful detection gets filtered. This additionally helps to avoid some of the misdetections that happen due to noise in the environment e.g. bad lighting conditions, reflections and temporary occlusions. Filtered sample output from the pipeline, showcasing the small target regions as well as both classes, can be found in fig. 5.

4 Experimental Results

We evaluate the performance of our developed pipeline in two stages. First the standalone performance of the trained neural networks is evaluated while in the second step we focus on the different pipelines with various combinations of the models. The results for the object detector training can be found in tab. 2. The metrics for the evaluation were taken from the documentation for the COCO challenges and calculated via tensorflow, focusing on average precision (AP) and average recall (AR) at different IoU thresholds and

target sizes.[1] The primary metric for the COCO challenges, ($AP_{0.50:0.95}$), averages over multiple IoU values at different thresholds (usually 10 thresholds with a stepsize of 0.05) to gather its results. We furthermore opted to present the outcomes for an IoU of 50% or above (AP_{50}), meaning that only the best prediction with the highest overlap (min. 50%) of the correct area is taken into account. The table clearly shows a significant decline in AP and AR once the target is categorized as small (AP_s, AR_s), meaning its area is less than 32x32 pixels, when compared to the results for medium-sized markers (AP_m, AR_m) which are encompassing an area between 32^2 and 96^2 pixels. Slicing the input however lead to improvements in both of these metrics for that specific category, with gains of up to almost 50%. This is quite significant for the success of the pipeline as most of the markers fall into this category. The amount of training steps has a negligible impact once a certain threshold is reached. A longer training might lead to diminishing returns or even overfitting, thus we stopped the process early. The EfficientDet neural network [10], which was tried for comparison, could not reach the performance of our developed models in every metric. The built Bayesian Neural Networks are shown in tab. 3. It becomes apparent that these mainly vary by the number of parameters, which is caused by different amounts of convolutional layers per model. The training and test data was shuffled, parts of it augmented and the models were trained using a custom loop with early stop functionality. The networks were then evaluated by using TensorFlow internal metrics as well as a manual evaluation step which is more representative of the actual usage of the model inside the processing pipelines. For the latter, inference was performed on 700 samples from the test dataset with 100 forward passes each, where the more likely prediction was taken as the result for every pass. This can then be seen as a sampling from the learned posterior distributions providing the aleatoric uncertainty estimation. Two example results from this process are displayed in figure 6. These evaluations are quite similar and strong for the first three BNNs from the table, with negligible differences between them. Even though the third model technically reaches the highest scores, there clearly is some overhead when compared to the two shallower ones. The different sets of tests deliver similar results when run on the same model, which confirms the applicability for the use case. It appears that the trained model is robust against changes in data. Besides the trained models themselves, the complete pipeline consisting of the object detectors, classifiers and object trackers to add context in the videos was also evaluated. The building blocks for pipeline 1 to 3 consist of the Faster-RCNN with

network	training steps	$AP_{0.5}$	$AP_{0.50:0.95}$	AP_s	AP_m	AR_s	AR_m
Faster RCNN	5k	0.73	0.41	0.40	0.68	0.54	0.70
	10k	0.76	0.45	0.44	0.83	0.59	0.85
	20k	0.76	0.48	0.47	0.78	0.59	0.80
Faster RCNN	10k	0.93	0.66	0.66	0.81	0.73	0.84
(cropped input)	20k	0.89	0.67	0.67	0.78	0.74	0.85
EfficientDet4	20k	0.70	0.36	0.36	0.67	0.49	0.70

Table 2: Evaluation metrics for the different object detectors, evaluated on the labeled test dataset. The metric used for the accuracy evaluation is the Average Precision AP_{IoU} with an IoU threshold to compare the label region with the predicted region. AP and AR are calculated for different label sizes.

		conv layers	Act BN			test		MXT		DOT	
model	epochs			params	auc	acc	loss	prec	recall	prec	recall
BNN 1	58	1	x	10k	0.96	0.97	0.14	0.98	0.93	0.98	0.95
BNN 2	64	2	x	55k	0.97	0.97	0.34	0.98	0.96	1	0.95
BNN 3	9	3	x	211k	0.98	0.98	0.1	0.98	0.973	0.99	0.97
BNN 4	13	3		211k	0.69	0.8	1.0	0.99	0.4	0.67	0.99
LeNet	7	3	x	61k	0.99	0.99	0.01	0.99	0.99	0.99	0.99

Table 3: Bayesian model scores, evaluated on test data. The evaluation metrics are calculated on both classes together and separately. The parameters of the networks is displayed for evaluation of complexity against the scores.

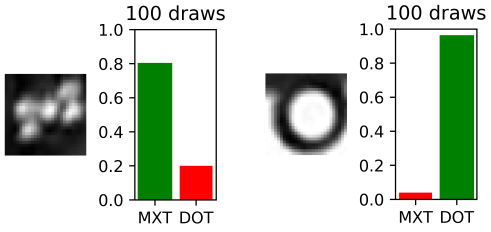

Figure 6: Example results obtained by the Bayesian Neural Network after 100 forward passes. On the left there are misclassifications on around 20% of the forward passes while the input on the right is correctly classified with less uncertainty.

20k training steps and, if needed, the third BNN model from table 3. For the last pipeline the object detectors were exchanged, using the Faster-RCNN trained on the cropped images (20k steps) to infer upon the sliced input frames. To perform the evaluation, single sample images were taken out of the test videos and then processed in isolation from the context of the video. These were then compared one-by-one to the same frame, taken from the filtered output of the pipeline after processing the entire clip. It can clearly be seen, that the additional information from the object trackers help to mitigate some of the short comings the object detector shows when it comes to small targets. The improvement in the underlying object detector model for the last pipeline directly translates to $10 - 20\%$ better recall for both target classes, which is way less than the 50% increases seen before when comparing only the trained models. This furthermore confirms the positive impact of the later parts of the pipeline.

		MXT		DOT	
pipeline	test env	precision	recall	precision	recall
1	isolated frame	0.99	0.67	0.97	0.7
	video context (filtered)	0.99	0.77	0.94	0.75
2	isolated frame	0.99	0.69	0.97	0.69
	video context (filtered)	0.99	0.81	0.95	0.76
3	isolated frame	0.99	0.71	0.96	0.70
	video context (filtered)	0.99	0.79	0.93	0.73
4	isolated frame	0.99	0.83	0.99	0.71
	video context (filtered)	0.98	**0.94**	0.99	**0.84**

Table 4: Results of different pipelines, evaluated on samples from test videos. This comparison shows the performances of the same frame, isolated and in the context of the video.

5 Discussion and Outlook

In this work we showcase how to develop a pipeline that efficiently combines state of the art object detection models with Bayesian Neural Networks to classify the detected objects. This approach leads to high performance object detections as well as an added security feature when estimating the aleatoric uncertainty of the model itself. Estimating this uncertainty helps to build trust into a machine learning model.

This approach can also be used in many different applications whenever small, seemingly hard to detect objects have to be tracked in highly dynamic environments. It shows that the pipeline, using contextual information, delivers constantly better results in comparison to isolated measurements. The filtering methods help to detect outliers, misclassifications and unstable object trackers, thereby cleaning the output from unwanted noise. As the detection of small objects still remains a challenge, even for modern machine learning models, this contribution once again confirms that a loss of information (e.g. resizing) leads to significantly worse results when it comes to this specific range of target sizes. The performance on medium sized target markers on the other hand is not affected by this in any relevant way.

Further work will focus on improving the recall for the DOT class, which is lacking behind its counterpart, as well as improving tracking and refining the model structure. One drawback of the improved metrics in the latest version of the Faster-RCNN model, and in turn the fourth pipeline, is a 5x increase in runtime when compared to its predecessors. Potential optimizations in this regard could be the skipping of certain grid boxes, as only few contain the sought after information at any given time. As the runtime aspect does not play a crucial role in our case, the focus was set on the maximization of the detection accuracy for the target markers, allowing for an accurate post-crash analysis based on videos. This seems to be achieved to a high degree when looking at the data presented in section 4, even though there still might be potential improvements to be found. Thanks to the cooperation with BMW we will be able to further try and evaluate the software on current crash tests provided by them. Data and code used to produce the results in this work can be accessed at `https://github.com/DataScienceLabFHSWF/crashtest_targetmarker_detection`. We thank the EG-30 at BMW for the cooperation and provision of additional data for the training of our models.

References

[1] COCO - Common Objects in Context. `https://cocodataset.org/#detection-eval`. [On-

line; accessed 2022-02-03].

[2] models/tf2_detection_zoo.md at master · tensorflow/models. https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md. [Online; accessed 2022-12-22].

[3] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, apr 2017. doi: 10.1080/01621459.2017.1285773. URL https://doi.org/10.1080%2F01621459.2017.1285773.

[4] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France, 07–09 Jul 2015. PMLR. URL https://doi.org/10.5555/3045118.3045290.

[5] G. Deodato, C. Ball, and X. Zhang. Bayesian Neural Networks for Cellular Image Classification and Uncertainty Analysis. preprint, Bioinformatics, Oct. 2019. URL https://doi.org/10.1101/824862.

[6] A. Harakeh, M. Smart, and S. L. Waslander. BayesOD: A Bayesian Approach for Uncertainty Estimation in Deep Object Detectors. *arXiv:1903.03838 [cs]*, Sept. 2019. URL https://doi.org/10.48550/arXiv.1903.03838. arXiv: 1903.03838.

[7] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors, 2016. URL https://doi.org/10.48550/arXiv.1611.10012.

[8] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014. URL https://doi.org/10.48550/arXiv.1312.61140.

[9] A. Lukezic, T. Vojír, L. Cehovin, J. Matas, and M. Kristan. Discriminative correlation filter with channel and spatial reliability. *Int J Comput Vis*, 126, 671–688, 2018. doi: 10.1007/s11263-017-1061-3. URL https://doi.org/10.1007/s11263-017-1061-3.

[10] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection. 2019. doi: 10.48550/ARXIV.1911.09070. URL https://arxiv.org/abs/1911.09070.