2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

# Uncertainty Estimation for Deep Neural Object Detectors in Safety-Critical Applications

Michael Truong Le[1] and Frederik Diehl[1] and Thomas Brunner[1] and Alois Knoll[2]

***Abstract*— Object detection algorithms are essential components for perceiving the environment in safety-critical systems like automated driving. However, current state-of-the-art algorithms based on deep neural networks can give high confidence values to falsely detected objects and it is therefore important to model uncertainty for these predictions.**

**In this paper, we propose two aleatoric uncertainty estimation algorithms for state-of-the-art deep learning based object detectors. Established algorithms for estimating uncertainty can either not be directly applied to object detection networks or result in high inference times. Instead, we adapt an existing method for aleatoric uncertainty estimation and propose another simple and efficient algorithm which is directly based on the multi-box detections. We show that these methods are able to assign high uncertainty values to false positives and visualize these in uncertainty maps. The uncertainty estimation methods are applied to a neural object detector and are compared with respect to their accuracy and inference time.**

## I. Introduction

Safety-critical systems utilizing camera-based object detection require reliable predictions. This means not only a high confidence accuracy but also the ability to produce a measure of confidence in these predictions, which can then be used to improve decision-making further down the line. Object detection algorithms have to overcome several difficulties which can cause false predictions, like lighting changes, object variations, smaller objects in the distance and occlusions.

In recent years, object detection algorithms have moved away from classical approaches—like HOG features [1] combined with Support Vector Machines or deformable part based models [2]—towards neural networks, where algorithms like Faster RCNN [3], SSD [4] or YOLO [5] have far surpassed classical methods in detection accuracy. Earlier stages of these models learn powerful representations by transforming the input space so that later layers are capable of effectively regressing several bounding boxes and classifying object categories for each instance. The multiple predictions for detected objects are then discarded by a well-known algorithm called non-maximum suppression (NMS) [8]. Even though their accuracy is greater compared to classical methods, these networks still make mistakes in the form of non-detected, wrongly classified or incorrectly detected objects. Furthermore, neural networks come with a disadvantage which is inherent to their training and structure: Because their predictions are produced by the nonlinear combination of millions of parameters, they are difficult to assess and their mispredictions are challenging to identify and eliminate.

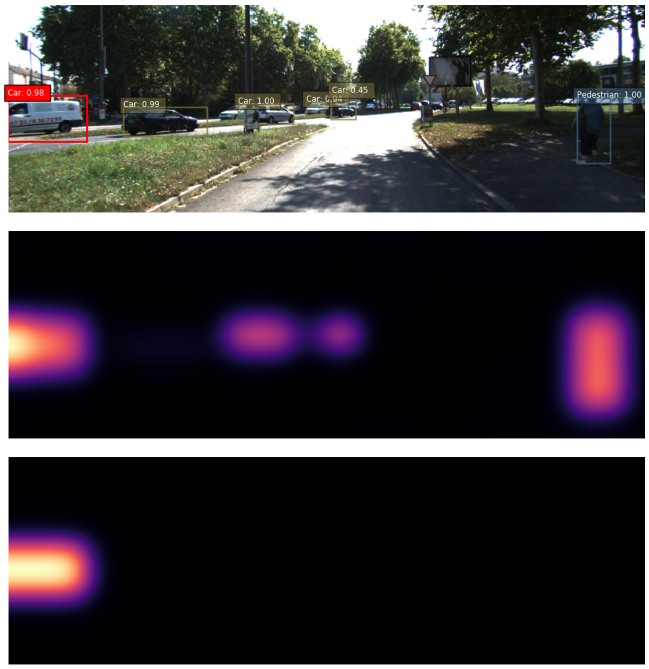


Fig. 1: Images quantitatively showing the results of the uncertainty estimation methods. From top to bottom the content of the images are: (i) Detections from a loss attenuated SSD in which false positives are marked in red, (ii) confidence uncertainties from loss attenuation and (iii) confidence uncertainties calculated using the redundancy method.

A neural network object detector produces a bounding box prediction, combined with a classification score that is often mistakenly believed to be a confidence probability. However, it is actually a normalized network output — usually a softmax activation. The networks tend to yield overconfident predictions which can lead to false positives with very high scores [6]. Current state-of-the-art object detectors cannot determine whether a prediction is a true or false positive. This is undesirable for safety-critical systems like autonomous driving in which a wrong detection can have fatal consequences. Therefore, it is not sufficient to rely on the classification score alone. In order to address this problem, the network can be extended to estimate an

[1]Michael Truong Le, Frederik Diehl and Thomas Brunner are with fortiss GmbH, affiliated institute of Technische Universität München, Munich, Germany

[2]Alois Knoll is with Robotics and Embedded Systems, Technische Universität München, Munich, Germany

additional value in the form of an uncertainty, which can then be used to supplement the normalized classification score and the regressed bounding box.

Uncertainty can be decomposed into two types, namely *epistemic* and *aleatoric* uncertainty. Kendall and Gal [7] argue that the latter is more important to model because epistemic uncertainty can be reduced with more training data. Epistemic uncertainty captures the ignorance of a predictive model itself and should therefore be large if a sample occurs that has not been seen by the model during the training process. It is therefore also called model uncertainty. Aleatoric uncertainty, on the other hand, describes the uncertainty in the world (for example caused by sensor noise or pixel discretization of the cameras). This type of uncertainty cannot be reduced with more data.

In this paper, we implement and compare two aleatoric uncertainty estimation methods for a safety critical application applied to the Multi-Box Single-Shot Detector. The first method is an adaptation of an existing algorithm for regression and classification, whereas the second method utilizes the multi-box proposals of current deep neural object detectors. We show that NMS can be utilized concurrently with multiple boxes to estimate aleatoric uncertainty. In this case, we use the KITTI dataset [9] to evaluate the uncertainty methods against the vanilla implementation of SSD.

## II. Related Work

Although deep neural networks have gained more and more attention in the last decades, there has been little active research in the area of uncertainty estimation for these models. There are two general methods on how uncertainty can be calculated: sampling-based and sampling-free calculations. Sampling-based calculations make several predictions for a single input (i.e. the same image), whereas sampling-free based methods are able to calculate the uncertainty with a single forward pass. The latter needs less computational resources but has not been well-studied and is mainly used for aleatoric uncertainties.

### A. Sampling-Based Methods

Estimating uncertainty for neural networks was studied in the 90's, primarily by using Bayesian Neural Networks (BNNs) [10], [11]. BNNs estimate epistemic uncertainty by placing a prior distribution over the weights of a neural network. This prior distribution is applied as additional parameters in form of a variance over the weights which are jointly optimized with the regular weight and biases in the training process. Recent research on optimizing the weight distributions has been focused on variational inference methods which use the ELBO (expected lower bound) of the Kullback-Leibler divergence as minimization objective. This is also called the variational free energy [12], [13]. Sampling weights from these distributions can intuitively be seen as sampling several instances of a network with the same neural architecture. Accordingly, epistemic uncertainty can be estimated from these models by forward-passing an input multiple times through the network and drawing a set of weights from the trained distribution for each pass. This results in several sample predictions for a single input, from which the variance can be computed.
Gal and Ghahramani [6] use dropout layers as an approximation to a Gaussian process to model epistemic uncertainty. Dropout is a technique which stochastically omits neurons in a layer, usually to avoid overfitting during the training process, and is normally turned off for inference [14]. However, for epistemic uncertainty estimation dropout is not deactivated during inference time but is used to re-run an input through the network, generating several predictions by randomly dropping neurons in the dropout layers for each forward pass. Just like in BNNs, several networks can be sampled and the epistemic uncertainty is calculated through the variance of the predictions made from the drawn weights. The inference time can be reduced by passing an input just once until it reaches the dropout layer, where it is then evaluated on the activated neurons. This technique is referred to as Monte-Carlo dropout (MC-dropout) since the posterior distribution is estimated through Monte-Carlo integration using dropout layers. Dropout is often used for fully-connected layers in a network and is usually not used in convolutional layers. In a follow-up work, MC-dropout was implemented for convolutional layers and showed improved results [15].
In contrast to the methods described up to this point, which are based on Bayes theory, Lakshminarayanan, Pritzel, and Blundell [16] propose a non-Bayesian method using an ensemble of neural networks. They utilize deep ensembles to calculate the variance by averaging the predictions of an input which has been passed through multiple networks. The networks of the ensemble are trained independently from each other on the whole dataset utilizing adversarial training produced by the fast gradient sign method [17].

All sampling-based uncertainty estimation methods suffer from the fact that an input needs to be reevaluated several times to get a good estimation of the uncertainty. This is unwanted because object detection networks already have high inference times. Furthermore, ensembles of networks need not only to rerun a single input but also need to store several sets of weights for each network.

### B. Sampling-Free Methods

One sampling-free method to estimate the aleatoric uncertainty is to add an additional variance output for each model output and changing the loss function according to

$$\mathscr{L}(\Theta) = \frac{||\mathbf{y} - f(\mathbf{x})||^2}{2\sigma(\mathbf{x})^2} + \frac{1}{2}\log(\sigma(\mathbf{x})^2). \qquad (1)$$

This modified objective function can be seen as loss attenuation and is mainly used for regression tasks. This loss was adapted for classification tasks and combined with BNNs to prove that both aleatoric and epistemic uncertainty can be calculated concurrently in a single network [7].

Another sampling-free method was proposed recently by Choi, Lee, Lim, and Oh [18]. They estimate aleatoric and epistemic uncertainty by putting a Gaussian Mixture model on the output of the network to predict a single input. These networks are called Gaussian Density Models [19]. By using the predictions of each Gaussian mixture components instead of single class scores this method is able to calculate both epistemic and aleatoric uncertainty. This method was used for training a shallow feed-forward neural network to learn the heading of a vehicle from a small input space. In contrast, neural network object detectors are normally applied to raw image data and have large output space which, based on our experiments, leads to non-convergence of the model.

## III. Uncertainty Estimation for Deep Neural Object Detectors

Current state-of-the-art object detection algorithms utilizing neural networks only provide a single output value for the bounding box and the class confidence. In most cases this is not desirable as networks i) tend to be overconfident, providing high confidence values to false positives, or ii) output a displaced bounding box. For this reason, either a calibrated class confidence for the classification or an uncertainty measure is needed as an indicator for assessment. This section will describe two aleatoric uncertainty estimation methods for deep learning based object detectors. The methods will exemplarily be implemented on the Single Shot Detector (SSD) [4], but can also be used on any other default-box/anchor-based deep learning object detection architecture.

### A. Single Shot Detector

SSD is an object detection meta-architecture that uses anchors as starting point for the detections [20]. Faster-RCNN also belongs to these types of architectures. Each of these detectors use base networks like VGG-16 [21], Inception [22]–[24] or ResNet [25], to extract features for regressing bounding boxes and classifying object categories. SSD puts several convolutional layers on top of these to generate smaller feature maps. At the very end of these feature maps class confidences are estimated and bounding box offsets are regressed with respect to the anchor positions. The anchors are predefined for each pixel in the feature maps and vary within size and aspect ratio. The different sizes of the feature maps and the anchors allow SSD to detect objects in different scales. Earlier layers are responsible for detecting smaller objects and later layers for larger objects.

### B. Loss Attenuation for Object Detection

Aleatoric uncertainty captures observation noise and is represented as a Gaussian likelihood. For object detection, the method is implemented by appending additional output vectors to each anchor. Specifically, each network output $\mathbf{y}$ is augmented by a second output $\sigma_{la}$, resulting in an additional $(n_c+4)*n_a$ convolution filter operations in the last stages of the SSD detector. $n_c$ denotes the number of classes and $n_a$ denotes the total number of anchors. For the four bounding box offsets uncertainties Eq. (1) is used and rephrased to

$$\mathscr{L}(\Theta) = \frac{1}{n_a}\sum_{a\in A}\frac{1}{2}\Phi(\hat{\mathbf{y}}, f_{ebox}(\mathbf{X},a))e^{-s_a} + \frac{1}{2}e^{s_a}, \qquad (2)$$

where $s_a := log(\sigma^2_{la,box})$, $\hat{\mathbf{y}}$ is an encoded groundtruth box, $\mu_{enc,a} = f_{ebox}(\mathbf{X},a)$ is the predicted encoded box with respect to image $\mathbf{X}$ and anchor box $a$. $\Phi(b_1,b_2)$ is a distance measure, in our case a smooth L1 function, between an encoded box $b_1$ and an encoded box $b_2$.
In Eq. (2), we estimate the logarithm of the variance to increase numerical stability and, contrary to the standard formulation, add an additional exponential function to the second term. To decode the bounding boxes together with their estimated variance we draw samples from $\mathscr{N}(\mu_{enc,a}, \sigma_{la,box})$ and decode these with respect to their anchors resulting in several decoded box samples from which the estimates can be calculated via mean and variance.

For the classification task, the uncertainty is estimated with a Monte Carlo integration by sampling the network outputs through the softmax function. Note that $f_{conf}(\mathbf{X})$ are logits before the softmax normalization function. This results in the following stochastic loss:

$$\hat{f}_i(\mathbf{X}, \sigma_{la,conf}) = f_{conf}(\mathbf{X}) + \varepsilon_i\sigma_{la,conf}, \qquad (3)$$

$$\mathscr{L}(\Theta) = \frac{1}{N}\sum_{i=0}^{i=N} softmax(\hat{f}_{conf_i}(\mathbf{X}, \sigma_{la,conf})), \qquad (4)$$

where $\varepsilon_i$ are samples drawn from a standard normal distribution $\mathscr{N}(0,1)$ and $N$ is the number of samples. In the original formulation, the sampling was passed through a rephrased softmax function which we found to be unnecessary.
By combining loss attenuation for bounding box offset regression and object classification it is now possible to estimate aleatoric uncertainty for the whole network output.

### C. Aleatoric Uncertainty Estimation via Redundancy

Even though aleatoric uncertainty estimation can be calculated via loss attenuation it still needs more parameters than the standard detector. In this section, we will describe a simple but efficient method to calculate the aleatoric uncertainty without any sampling.

Current state-of-the-art object detectors already produce a set of object observations $O^k_1, O^k_2, \dots O^k_{B_k}$ for the classification and their bounding boxes in which B are the number of observations for a particular object and $k \in K$ are the number object proposals of an object detector before the NMS step. The vector $O_i$ consists of the confidence measurements for all classes and four values for the bounding box offsets. Assuming the observations to be independent and identically distributed the sampling mean $E[\hat{\mathbf{O}}_i] = \hat{\mu}_{\mathbf{re}}$ and the sampling variance $Var[\hat{\mathbf{O}}_i] = \sigma_{re}$ of this distribution for a particular object are computed as in Eq. (5):

$$\hat{\mu}_{re} = \frac{1}{B}\sum_{i=0}^{i=B} O_i, \quad \sigma_{re} = Var[\hat{\mathbf{O}}_i] = \frac{Var[O]}{B}. \qquad (5)$$

With this prior knowledge, the aleatoric uncertainty can be calculated by iterating all detection proposals of the anchors. For each proposal, the Jaccard overlap will be computed against all other proposals. Only the observations $O_i^k$ with a Jaccard overlap bigger than a certain threshold $\theta$ are counted as a match. After this matching, step there are a total of $B_k$ boxes for the proposal $k$ which are then used to estimate the uncertainty.
This method can be run concurrently with NMS so that the overall additional computational costs are minimized. We also point out that sampling is not needed here because current state-of-the-art object detectors themselves produce a set of observations by detecting objects for each anchor box.

## IV. Evaluation

In this section, we present the results of the uncertainty estimation. The two methods were evaluated using the KITTI dataset which contains over 7000 images of labeled traffic participants including cars, pedestrians, trams, vans, trucks and cyclists. The detectors used were a standard SSD, on which we performed the redundancy method, and an enhanced SSD with additional variance output for loss attenuation. Both detectors are implemented and trained on the tensorflow object detection API [20]. We show that using the uncertainties and simple thresholding can reduce the number of false positives.

### *A. Training*

For training the SSD network to estimate aleatoric uncertainty we used an Inception V2 as a base network from which we extracted *mixed2* and *mixed7* layers to build the SSD architecture. The Inception V2 was pretrained on the COCO dataset [26]. We used the standard parameters for the SSD network which are described in their respective paper with a batch size of 16. We used the rmsprop optimizer [27] with an initial learning rate of 0.001 and a decaying factor of 0.95. We also apply gradient clipping with a value of 1 so that the classification attenuated loss does not result in high values in earlier stages of the training. The attenuated classification loss was calculated with 100 samples which were sufficient for the network to be optimized well according to the log variance output.

### *B. Visualized Uncertainty Results*

For visualizing the class uncertainties each prediction (including backgrounds) is taken into account. We start with an empty uncertainty image filled with zeros. Afterwards, for each prediction, the corresponding class variance is compared to the values inside the uncertainty image bounded by the predicted box. If there are pixels which are smaller than the variance value, these values will be substituted by the variance of the current prediction. The image is then smoothed with a Gaussian filter.

The visualization results can be seen in Fig. 1. The top image shows predicted objects—several cars and a pedestrian—with their confidence scores. The uncertainties calculated by

TABLE I: False and true positives from the KITTI validation dataset. For each method, we listed the true and false positives before (top rows) and after (middle rows) the cut-off. To get better indications on the performance the inverse ratio between these numbers was calculated.

| Classes | | Car | Van | Truck | Pedestrian | Cyclist | Tram |
|---|---|---|---|---|---|---|---|
| Redundancy | tp | 1703 | 185 | 48 | 194 | 84 | 28 |
| | | 1524 | 168 | 46 | 181 | 78 | 28 |
| | | 0.90 | 0.90 | 0.96 | 0.93 | 0.93 | 1.00 |
| | fp | 64 | 23 | 4 | 28 | 13 | 1 |
| | | 58 | 20 | 3 | 28 | 13 | 0 |
| | | 0.91 | 0.87 | 0.75 | 1.00 | 1.00 | 0.00 |
| Attenuated | tp | 1708 | 222 | 51 | 196 | 96 | 29 |
| | | 1687 | 213 | 51 | 195 | 92 | 29 |
| | | 0.99 | 0.96 | 1.00 | 0.99 | 0.96 | 1.0 |
| | fp | 84 | 26 | 3 | 71 | 28 | 2 |
| | | 78 | 22 | 3 | 60 | 23 | 1 |
| | | 0.93 | 0.856 | 1.00 | 0.85 | 0.82 | 0.5 |

the loss attenuation and the redundancy method are depicted in the middle and bottom image respectively. In this case, the redundancy method was applied to the loss attenuated trained network so that both methods can be compared to each other.

The highest uncertainty values are located on the leftmost object in the image. This car is a false positive with a high score (0.98) whose real label is a *van*. Furthermore, it can be seen that the redundancy method is dependent on the number of boxes. If there are no overlapping boxes uncertainty values cannot be estimated which is the case for very small objects (the small cars in the center of the image). On the other hand, the loss attenuated learned uncertainty is able to output a value for each of the objects but is significantly slower (see IV-D). To further assess the ability of these methods for assigning false positively detected objects high uncertainty values we propose a statistical analysis in the next section.

### *C. Uncertainty Estimation Results*

We validated the performance of the detectors on the first 748 images from the KITTI dataset on which we performed a cut-off for detections with a high uncertainty. Table I shows the performance with respect to the true and false positives. The ratio between the number of false/true positives after and before the thresholding can be seen in the gray rows. Relatively, both methods are able to eliminate more false positives than true positives, except for cars in the redundancy method. The redundancy method performs not as well as the attenuated method because it loses more true positives while retaining more false positives. This fact can also be seen in Table II, which shows the mean uncertainty values for false and true positives. For the loss attenuation method, the average difference in uncertainty for false positives is four times higher than for true positives compared to the three times difference of the redundancy method. The effect of this difference can be especially seen in the car-category in the first table. Loss attenuation is able to retain almost all true positives and still reduce the number of false positives.

TABLE II: Mean uncertainty values for false and true positives of the KITTI validation dataset.

| Method | Redundancy | Loss Attenuation |
|---|---|---|
| False Positives | 0.061 | 0.125 |
| True Positives | 0.026 | 0.033 |

## D. *Inference Time Comparison*

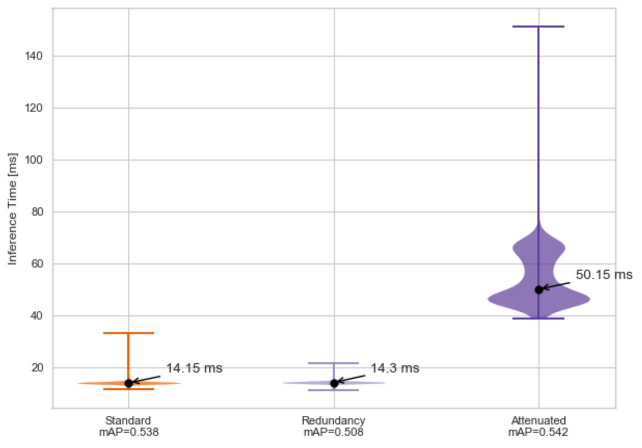
Fig. 2 shows the inference time for each method. Although the loss attenuated trained network performs best with respect to the mAP it has by far the highest inference time with a median of 50.15ms and with outliers up to over 150ms. The cause for the high inference time are the additional parameters needed in the network for estimating the uncertainty which depends on the number of anchors.
Due to the fact that the redundancy method is applied to the post-processing step of the detection pipeline the inference times of the detectors are almost equal. On the other hand, the median NMS inference time of the redundancy method (1.41ms) is more than double the value of the standard method (0.62ms) with outliers reaching to over 6ms. This is because the NMS-step is enhanced by calculating the uncertainties and is therefore depending on the number of detected objects.

## V. Discussion

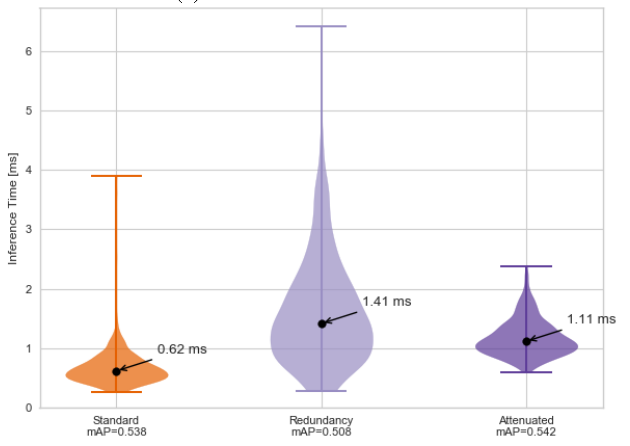
We showed that both methods are able to assign large uncertainty values for false positives. By thresholding the detections with respect to their estimated uncertainties it is possible to eliminate more false positives than true positives. By varying the threshold, we can trade-off mAP for false positives. Although not discussed in this paper, bounding box uncertainties are concurrently estimated with the classification uncertainty in both methods which can be used for further processing in a perception pipeline.

Uncertainty estimated through redundancy depends on multiple predictions per object. For few or no overlaps, the variance estimation becomes inaccurate or even results in zero if no other box overlaps. This applies to very large or small objects in the image. It is known that these can be optimized by setting appropriate anchors defined by the aspect ratios and sizes [4]. The anchors can then be tuned for a specific task so that the network can regress multiple boxes for one object. We will leave this task open for future work.

In contrast, loss attenuation provides uncertainty measures for all predicted boxes but needs to be trained with the network itself. In our experiments, we have seen that training can be unstable because of the classification loss. To circumvent this we have clipped the gradient value.
Using loss attenuation also improves the mAP of the detector but results in triple the inference time and double the amount of weights in the output layers.



(a) Detector Inference Times



(b) NMS Inference Times

Fig. 2: Violin plots showing the inference times [ms] of the detector (a) and the NMS post-processing step (b). Three methods are depicted: Standard for the vanilla SSD network, redundancy for the redundancy method applied on the standard method and attenuated for the loss attenuation. The black dots represent the median of the distributions and the whiskers the outliers.

## VI. Conclusion

In this paper, we presented uncertainty estimation methods for deep neural object detectors which are able to estimate the variance of the classification score and the bounding boxes. We evaluated the methods on the basis of mean average precision and the changes in true positives and false positives. It has been shown, that, on average, both methods are able to assign higher uncertainty values to false positives compared to true positives. The loss attenuation method produces uncertainties for every box prediction but requires additional weights and more inference time. In contrast, the redundancy method, while significantly faster, cannot produce uncertainties for very small or very large objects.

We believe the two methods proposed herein can form a useful part of the object detection arsenal, particularly in safety-critical applications such as automated driving.

## References

[1] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 2005, pp. 886–893.

[2] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, no. 9, pp. 1627–1645, 2010.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 91–99.

[4] W. Liu, D. Anguelov, D. Erhan, *et al.*, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*, Springer, 2016, pp. 21–37.

[5] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You Only Look Once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 2016, pp. 779–788.

[6] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of The 33rd International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2016, pp. 1050–1059.

[7] A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In *Advances in Neural Information Processing Systems*, 2017, pp. 5580–5590.

[8] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *Proceedings of the International Conference on Pattern Recognition*, IEEE Computer Society, 2006, pp. 850–855.

[9] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[10] J. Denker and Y. Lecun, "Transforming Neural-Net Output Levels to Probability Distributions," in *Advances in Neural Information Processing Systems*, Morgan Kaufmann, 1991, pp. 853–859.

[11] R. M. Neal, *Bayesian learning for neural networks*, PhD Thesis, 1995.

[12] A. Graves, "Practical variational inference for neural networks," in *Advances in Neural Information Processing Systems*, Curran Associates Inc., 2011, pp. 2348–2356.

[13] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, JMLR, 2015, pp. 1613–1622.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, no. 1, pp. 1929–1958, 2014.

[15] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with bernoulli approximate variational inference," *arXiv:1506.02158*, 2015.

[16] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, 2017, pp. 6405–6416.

[17] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv:1412.6572*, 2014.

[18] S. Choi, K. Lee, S. Lim, and S. Oh, "Uncertainty-Aware Learning from Demonstration using Mixture Density Networks with Sampling-Free Variance Modeling," *arXiv:1709.02249*, 2017.

[19] C. M. Bishop, "Mixture density networks," 1994.

[20] J. Huang, V. Rathod, C. Sun, *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 2017, pp. 3296–3297.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, 2014.

[22] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[23] C. Sdzegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 2016, pp. 2818–2826.

[24] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Conference on Artificial Intelligence*, AAAI Press, 2017, pp. 4278–4284.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 2016, pp. 770–778.

[26] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, *Microsoft COCO: Common Objects in Context*, 2014.

[27] T. Tieleman and G. Hinton, *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*, COURSERA: Neural Networks for Machine Learning, 2012.