Single Shot MC Dropout Approximation

Kai Brach [1] **Beate Sick** [2] **Oliver Dürr** [3]

Abstract

Deep neural networks (DNNs) are known for their high prediction performance, especially in perceptual tasks such as object recognition or autonomous driving. Still, DNNs are prone to yield unreliable predictions when encountering completely new situations without indicating their uncertainty. Bayesian variants of DNNs (BDNNs), such as MC dropout BDNNs, do provide uncertainty measures. However, BDNNs are slow during test time because they rely on a sampling approach. Here we present a single shot MC dropout approximation that preserves the advantages of BDNNs without being slower than a DNN. Our approach is to analytically approximate for each layer in a fully connected network the expected value and the variance of the MC dropout signal. We evaluate our approach on different benchmark datasets and a simulated toy example. We demonstrate that our single shot MC dropout approximation resembles the point estimate and the uncertainty estimate of the predictive distribution that is achieved with an MC approach, while being fast enough for real-time deployments of BDNNs.

1. Introduction

Over the last, decade deep neural networks (DNN) have arisen as the dominant technique for the analysis of perceptual data. Also in safety-critical applications like autonomous driving, where the vehicle must be able to understand its environment, DNNs have seen rapid progress in several tasks (Grigorescu et al., 2019).

However, classical DNNs have deficits in capturing the model uncertainty (Kendall & Gal, 2017),(Gal & Ghahramani, 2016). But when using DNN models in safety-critical applications, it is mandatory to provide an uncertainty measure that can be used to identify unreliable predictions (Michelmore et al., 2018) (Feng et al., 2018) (Harakeh et al., 2019) (Miller et al., 2018) (McAllister et al., 2017).

For example, in the field of robotics (Sünderhauf et al., 2018), medical applications, or autonomous driving (Bojarski et al., 2016), where machines interact with humans, it is important to identify situations where a model prediction is unreliable and a human intervention is necessary. This can, for example, be situations which are completely different from all that occurred during training.

Employing Bayesian DNNs (BDNNs) (MacKay, 1992) tackles the problem and allows to compute an uncertainty measure. However, state of the art BDNNs require sampling during deployment leading to computation times that are by the factor of MC runs larger than a classical DNNs. This work overcomes this drawback by providing a method that allows to approximate the expected value and variance of a BDNN's predictive distribution in a single run. It has therefore the same computation time as a classical DNN. We focus here on a special variant of BDNNs which is known as MC dropout (Gal & Ghahramani, 2016). While our approximation method is applicable also to convolutional neural networks and classification settings, we focus in this work on regression through fully connected networks.

Ensembling based models take an alternative approach to estimate uncertainties and have been successfully applied to DNNs (Lakshminarayanan et al., 2017; Pearce et al., 2020). But ensemble methods do also not allow to quantify the uncertainty in a single shot manner.

2. Related Work

2.1. MC Dropout Bayesian Neural Networks

BDNNs are probabilistic models that capture the uncertainty by means of probability distributions. Probabilistic DNNs, which are non-Bayesian, only define a distribution for the conditional outcome. In common probabilistic DNNs the output nodes are controlling the parameters of a conditional probability distribution (CPD) of the outcome. For regression type problems a common choice for the CPD is the normal distribution $N(\mu, \sigma^2)$, where the variance σ^2 quantifies the data uncertainty, known as aleatoric un-

[*]Equal contribution [1]Elektrobit Automotive GmbH, Germany [2]IDP, Zurich University of Applied Sciences, Switzerland, and EBPI, University of Zurich, Switzerland [3]IOS, Konstanz University of Applied Sciences, Germany. Correspondence to: Kai Brach <kai.brach@elektrobit.com>, Oliver Dürr <oliver.duerr@htwg-konstanz.de>, Beate Sick <sick@zhaw.ch>.

Presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning.

certainty. BDNNs define in addition distributions for the weights which translate in a distribution of the modeled parameters. In this manner the model uncertainty is captured, which is known as epistemic uncertainty (Der Kiureghian & Ditlevsen, 2009). In case of MC dropout BDNNs each weight distribution is a Bernoulli distribution: the weight takes with the dropout probability p^* the value zero and with probability $1 - p^*$ the value w. All weights starting from the same neuron are set to zero simultaneously. The dropout probability p^* is usually treated as a fixed hyperparameter and the weight-value w is tuned during the training.

In contrast to standard dropout (Srivastava et al., 2014), the weights in MC dropout are not frozen and rescaled after training, but the dropout procedure is also done during test time. It can be shown that MC dropout is an approximation to a BDNN (Gal & Ghahramani, 2016). MC dropout BDNNs were successfully used in many applications and have proven to yield improved prediction performance and allow to define uncertainty measures to identify individual unreliable predictions (Gal & Ghahramani, 2016), (Ryu et al., 2019), (Dürr et al., 2018), (Kwon et al., 2020). To employ a trained Bayesian DNN in practice one performs several runs of predictions. In each run, weights are sampled from the weight distributions leading to a certain constellation of weight values that are used to compute the parameters of a CPD. To determine the outcome distribution of a BDNN, we draw samples from the CPDs that resulted from different MC runs. In this way, the outcome distribution incorporates the epistemic and aleatoric uncertainty. A drawback of a MC dropout BDNN compared to its classical DNN variant is the increased computing time. The sampling procedure leads to a computing time that is prohibitive for many real-time applications like autonomous driving.

2.2. Moment Propagation

Our method relies on statistical moment propagation (MP). More specifically, we propagate the expectation and the variance, of our signal distribution through the different layers of a neural network. The variance of the signal arises due to the dropout process. Quantifying the variance after a transformation is also done in error propagation (EP). EP quantifies how an uncertainty of an input which is transformed by a function (i.e. a measurement error) transfers to an uncertainty of the output of this function. In case of a continuous output it is common to characterize the uncertainty by the variance. This approach is also used in statistics as the delta method (Dorfman, 1938). In MP we approximate the layer-wise transformations of the variance and the expected value. A similar approach has also been used for neural networks before (Frey & Hinton, 1999; Adachi, 2019), and used to detect adversarial examples in (Jin, 2015) and (Gast & Roth, 2018).

But, due to our best knowledge, our approach is the first method that provides a single shot approximation to the expected value and the variance of the predictive distribution resulting from a MC dropout NN.

3. Methods

The goal of our method[1] is to approximate the expected value E and the variance V of the predicted output which is obtained by the above described MC dropout method. When propagating an observation through a MC dropout network, we get each layer with p nodes an activation signal with an expected value E (of dimension p) and a variance given by a variance-covariance matrix V (of dimension $p \times p$). We neglect the effect of correlations between different activations, which are small anyway in deeper layers due to the decorrelation effect of the dropout. Hence, we only consider diagonal terms in the correlation matrix. In the following, we describe for each layer-type in a fully connected network how the expected value E and its variance V is propagated. As layer-type we consider dropout, dense, and ReLU activation layer. Figure 1 provides an overview of the layer-wise abstraction.

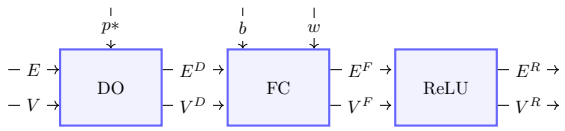

Figure 1. Overview of the proposed method. The expectation E and V flow through different layers of the network in a single forward pass. Shown is an example configuration in which Dropout (DO) is followed by Dense (FC) and a ReLU activation. More complex networks can be build by different arrangements of the individual blocks.

3.1. Dropout Layer

We start our discussion, with the effect of MC dropout. Let E_i be the expectation at the i'th node of the input layer and V_i the variance at the i'th node. In a dropout layer the random value of a node i is multiplied independently with a Bernoulli variable $Y \sim \texttt{Bern}(\texttt{p}^*)$ that is either zero or one. The expectation E_i^D of the i'th node after dropout is then given by:

$$E_i^D = E_i(1 - p^*) \tag{1}$$

For computing the variance V_i^D of the i'th node after dropout, we use the fact that the variance $V(X \cdot Y)$ of the product of two independent random variables X and Y, is given by (Goodman, 1960):

[1] https://github.com/kaibrach/Moment-Propagation

$$V(X \cdot Y) = V(X)V(Y) + V(X)E^2(Y) + E^2(X)V(Y) \tag{2}$$

With $V(Y) = p^*(1-p^*)$, we get:

$$V_i^D = V_i \cdot p^*(1-p^*) + V_i(1-p^*)^2 + E_i^2 \cdot p^*(1-p^*) \tag{3}$$

Dropout is the only layer in our approach where uncertainty is created. I.e. even if the input has $V_i = 0$ the output of the dropout layer has $V_i^D > 0$ for $p^* \neq 0$.

3.2. Dense Layer

For the dense layer with p input and q output nodes, we compute the value of the i'th output node as $\sum_j^p w_{ji}x_j + b_i$, where $x_j, j = 1 \dots p$ are the values of the input nodes. Using the linearity of the expectation, we get the expectation E_i^F of the i'th output node from the expectations, $E_j^F, j = 1 \dots p$, of the input nodes:

$$E_i^F = \sum_{j=1}^{p} w_{ji}E_j + b_i \tag{4}$$

To calculate the change of the variance, we use the fact that the variance under a linear transformation behaves like $V(w_{ji} \cdot x_j + b) = w_{ji}^2 V(x_j)$. Further, we assume independence of the j different summands, yielding:

$$V_i^F = \sum_{j=1}^{p} w_{ji}^2 V_j \tag{5}$$

3.3. ReLU Activation Layer

To calculate the expectation E_i^R and variance V_i^R of the i'th node after a ReLU, as a function of the E_i and V_i of this node before the ReLU, we need to make a distributional assumption. We assume that the input is Gaussian distributed, with $\phi(x) = N(x; E_i, V_i)$ the PDF, and $\Phi(x)$ the corresponding CDF, we get (see (Frey & Hinton, 1999) for a derivation) for the expectation and variance of the output:

$$E_i^R = E_i \cdot \Phi\left(\frac{E_i}{\sqrt{V_i}}\right) + \sqrt{V_i} \cdot \phi\left(\frac{E_i}{\sqrt{V_i}}\right) \tag{6}$$

$$V_i^R = (E_i^2 + V_i)\Phi\left(\frac{E_i}{\sqrt{V_i}}\right) + E_i\sqrt{V_i} \cdot \phi\left(\frac{E_i}{\sqrt{V_i}}\right) - E_i^{R2} \tag{7}$$

4. Results

4.1. Toy Dataset

We first apply our approach to a one dimensional regression toy dataset, with only one input feature. We use a fully connected NN with three layers each with 256 nodes, ReLU activations and dropout after the dense layers. We have a single node in the output layer which is interpreted as the expected value μ of the conditional outcome distribution $p(y|x)$. We train the network using the MSE loss and apply dropout with $p^* = 0.3$. From the MC dropout BDNN, we get at each x-position $T = 30$ MC samples $\mu_t(x)$ from which we can estimate the expectation E_μ by the average value and V_μ by the variance of $\mu_t(x)$. For comparison, we use our MP approach to also approximate the expected value E_μ and the variance V_μ of μ at each x-position (see upper panel of 2). We also included the deterministic output $\mu(x)$ of the DNN in which dropout has only been used only during training. All three approaches yield nearly identical results, within the range of the training data. We attribute this to the fact, that we have plenty of training data and so the epistemic uncertainty is neglectable. In the lower panel of figure 2 a comparison of the uncertainty of $\mu(x)$ is shown by displaying an interval given by the expected value of $\mu(x)$ plus-minus two times the standard deviation of $\mu(x)$. Here the width of the resulting intervals of a BDNN via the MP approach and the MC dropout are comparable (the DNN has no spread). This indicates the usefulness of this approach for epistemic uncertainty estimation.

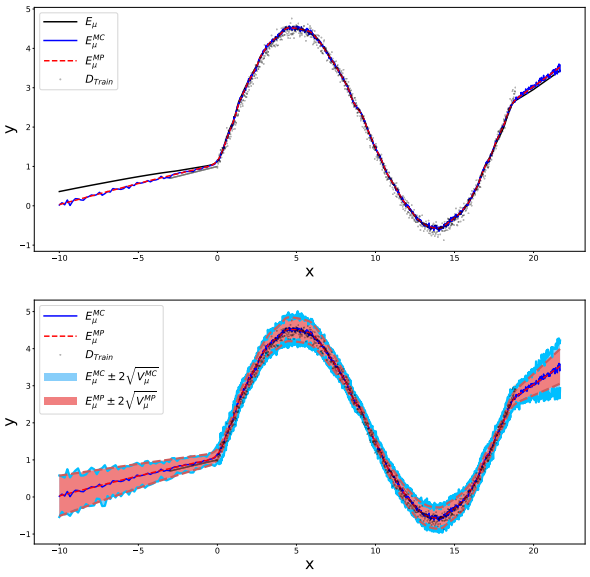

Figure 2. Comparison of the MP and MC dropout results of a BDNN and the results of a DNN. The NNs were fitted on train data that were available in the range of -3 to 19. In the upper panel the estimated expectations of the MC BDNN, the MP BDNN, and the DNN are compared. In the lower panel the predicted spread of $\mu(t)$ is shown for the MC and MP method.

4.2. UCI-Datasets

To benchmark our method, we redo the analysis of (Gal & Ghahramani, 2016) for the UCI regression benchmark

Table 1. Comparison of the average prediction performance in test RMSE (Root-Mean-Square Error), test NLL (Negative Log-Likelihood) and test RT (Runtime) including ± standard error on UCI regression benchmark datasets between MC and MP. *N* and *Q* correspond to the dataset size and the input dimension. For all test measures, smaller means better.

DATASET	*N*	*Q*	TEST RMSE		TEST NLL		TEST RT [S]	
			MC	MP	MC	MP	MC	MP
BOSTON	506	13	3.14 ±0.20	3.10 ±0.20	2.57 ±0.07	2.56 ±0.08	2.51 ±0.03	0.04 ±0.00
CONCRETE	1,030	8	5.46 ±0.12	5.40 ±0.12	3.12 ±0.02	3.13 ±0.03	3.37 ±0.04	0.04 ±0.00
ENERGY	768	8	1.65 ±0.05	1.61 ±0.05	1.95 ±0.04	2.01 ±0.04	2.84 ±0.03	0.04 ±0.00
KIN8NM	8,192	8	0.08 ±0.00	0.08 ±0.00	-1.10 ±0.01	-1.11 ±0.01	7.37 ±0.06	0.04 ±0.00
NAVAL	11,934	16	0.00 ±0.00	0.00 ±0.00	-4.36 ±0.01	-3.60 ±0.01	9.69 ±0.11	0.04 ±0.00
POWER	9,568	4	4.05 ±0.04	4.04 ±0.04	2.82 ±0.01	2.84 ±0.01	6.85 ±0.07	0.04 ±0.00
PROTEIN	45,730	9	4.42 ±0.03	4.41 ±0.02	2.90 ±0.00	2.91 ±0.00	31.38 ±0.09	0.05 ±0.00
WINE	1,599	11	0.63 ±0.01	0.63 ±0.01	0.95 ±0.01	0.95 ±0.01	4.78 ±0.01	0.04 ±0.00
YACHT	308	6	2.93 ±0.22	2.91 ±0.26	2.35 ±0.07	2.11 ±0.07	2.01 ±0.01	0.04 ±0.00

dataset. We use the same NN model as Gal and Ghahramani, which is a fully connected neural network including one hidden layer with ReLU activation in which the CPD $p(y|x)$ over $T = 10,000$ MC runs is given by sampling from the normal PDF:

$$p(y|x) = \frac{1}{T} \sum_t N(y; \mu_t(x), \tau^{-1}) \tag{8}$$

Again $\mu_t(x)$ is the single output of the BDNN for the t'th MC run. To derive a predictive distribution Gal assumes in each run a Gaussian distribution, centered at μ and a precision τ, corresponding to the reciprocal of the variance. The parameter μ is received from the NN and τ is treated as as a hyperparameter. For the MP model, the MC sampling (Eq. 8) is replaced by integration:

$$\begin{aligned} p(y|x) &= \int N(y; \mu', \tau^{-1}) N(\mu'; E^{\text{MP}}, V^{\text{MP}}) \, d\mu' \\ &= N(y; E^{\text{MP}}, V^{\text{MP}} + \tau^{-1}) \end{aligned} \tag{9}$$

We used the same protocol as (Gal & Ghahramani, 2016) which can be found at https://github.com/yaringal/DropoutUncertaintyExps. Accordingly, we train the network for 10× the epochs provided in the individual dataset configuration. As described in (Gal & Ghahramani, 2016) an excessive grid search over the dropout rate $p^* = 0.005, 0.01, 0.05, 0.1$ and different values of the precision τ is done. The hyperparameters minimizing the validation NLL are chosen and applied on the testset.

We report in table 1 the test performance (RMSE and NLL) achieved via MC BDNN using the optimal hyperparameters for the different UCI datasets. We also report the test RMSE and the NLL achieved with our MP method. Allover, the MC and MP approaches produces similar results. However, as shown in the last column in the table the MP method is much faster, having only to perform one forward pass instead of $T = 10,000$ forward passes.

5. Discussion

With our MP approach we have introduced an approximation to MC dropout which requires no sampling but instead propagates the expectation and the variance of the signal through the network. This results in a time saving by a factor that approximately corresponds to the number of MC runs (in our benchmark experiment 10,000). We have shown that our fast MP approach approximates precisely the expectation and variance of the prediction distribution achieved by MC dropout. Also the achieved prediction performance in terms of RMSE and NLL do not show significant differences when using MC dropout or our MP approach. Hence, our presented MP approach opens the door to include uncertainty information in real-time applications.

We are currently working on extending the approach to different architectures such as convolutional neural networks.We are also investigating how to make use of the uncertainty information to detect novel classes in classification settings.

6. Acknowledgements

We are very grateful to Elektrobit Automotive GmbH for supporting this research work. Further, part of the work has been founded by the Federal Ministry of Education and Research of Germany (BMBF) in the project DeepDoubt (grant no. 01IS19083A).

References

Adachi, J. Estimating and factoring the dropout induced distribution with gaussian mixture model. In *International Conference on Artificial Neural Networks*, pp. 775–792. Springer, 2019.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. End to End Learning for Self-Driving Cars. apr 2016. URL http://arxiv.org/abs/1604.07316.

Der Kiureghian, A. and Ditlevsen, O. Aleatory or epistemic? Does it matter? *Structural safety*, 31(2):105–112, 2009.

Dorfman, R. A. A note on the! d-method for finding variance formulae. *Biometric Bulletin*, 1938.

Dürr, O., Murina, E., Siegismund, D., Tolkachev, V., Steigele, S., and Sick, B. Know When You Don't Know: A Robust Deep Learning Approach in the Presence of Unknown Phenotypes. *Assay and drug development technologies*, 16(6):343–349, 2018.

Feng, D., Rosenbaum, L., and Dietmayer, K. Towards Safe Autonomous Driving: Capture Uncertainty in the Deep Neural Network for Lidar 3D Vehicle Detection. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2018-Novem:3266–3273, 2018. doi: 10.1109/ITSC.2018.8569814.

Frey, B. J. and Hinton, G. E. Variational learning in nonlinear gaussian belief networks. *Neural Computation*, 11(1):193–213, 1999. ISSN 08997667. doi: 10.1162/089976699300016872. URL https://ieeexplore.ieee.org/abstract/document/6790581/.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *33rd International Conference on Machine Learning, ICML 2016*, volume 3, pp. 1651–1660. International Machine Learning Society (IMLS), 2016. ISBN 9781510829008.

Gast, J. and Roth, S. Lightweight Probabilistic Deep Networks. Technical report, 2018.

Goodman, L. A. On the Exact Variance of Products. *Journal of the American Statistical Association*, 55 (292):708–713, may 1960. ISSN 01621459. doi: 10.2307/2281592. URL http://www.jstor.org/stable/2281592http://www.cs.cmu.edu/$\sim$cga/var/2281592.pdf.

Grigorescu, S., Trasnea, B., Cocias, T., and Macesanu, G. A Survey of Deep Learning Techniques for Autonomous Driving. *Journal of Field Robotics*, 37(3): 362–386, oct 2019. doi: 10.1002/rob.21918. URL http://arxiv.org/abs/1910.07738http://dx.doi.org/10.1002/rob.21918.

Harakeh, A., Smart, M., and Waslander, S. L. BayesOD: A Bayesian Approach for Uncertainty Estimation in Deep Object Detectors. 2019. URL http://arxiv.org/abs/1903.03838.

Jin, J. Robust Convolutional Neural Networks under Adversarial Noise. *CoRR*, abs/1511.0, 2015. URL http://arxiv.org/abs/1511.06306.

Kendall, A. and Gal, Y. What uncertainties do we need in Bayesian deep learning for computer vision? Technical report, 2017. URL http://papers.nips.cc/paper/7141-what-uncertainties-do-we-need.

Kwon, Y., Won, J.-H., Kim, B. J., and Paik, M. C. Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation. *Computational Statistics & Data Analysis*, 142: 106816, 2020.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pp. 6402–6413, 2017.

MacKay, D. J. C. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3): 448–472, may 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.3.448.

McAllister, R., Gal, Y., Kendall, A., Van Der Wilk, M., Shah, A., Cipolla, R., and Weller, A. Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning. In *IJCAI International Joint Conference on Artificial Intelligence*, 2017. ISBN 9780999241103. doi: 10.24963/ijcai.2017/661.

Michelmore, R., Kwiatkowska, M., and Gal, Y. Evaluating Uncertainty Quantification in End-to-End Autonomous Driving Control. nov 2018. URL http://arxiv.org/abs/1811.06817.

Miller, D., Nicholson, L., Dayoub, F., and Sunderhauf, N. Dropout Sampling for Robust Object Detection in Open-Set Conditions. In *Proceedings - IEEE International Conference on Robotics and Automation*, 2018. ISBN 9781538630815. doi: 10.1109/ICRA.2018.8460700.

Pearce, T., Leibfried, F., Brintrup, A., Zaki, M., and Neely, A. Uncertainty in neural networks: Approximately bayesian ensembling. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020*, 2020.

Ryu, S., Kwon, Y., and Kim, W. Y. A Bayesian graph convolutional network for reliable prediction of molecular properties with uncertainty quantification. *Chemical Science*, 10(36):8438–8446, 2019.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014. ISSN 15337928.

Sünderhauf, N., Brock, O., Scheirer, W., Hadsell, R., Fox, D., Leitner, J., Upcroft, B., Abbeel, P., Burgard, W., Milford, M., and Corke, P. The limits and potentials of deep learning for robotics. *International Journal of Robotics Research*, 2018. ISSN 17413176. doi: 10.1177/0278364918770733.