

# Sampling-free Epistemic Uncertainty Estimation Using Approximated Variance Propagation

Janis Postels

Technical University Munich

[janis.postels@tum.de](mailto:janis.postels@tum.de)

Huseyin Coskun

Technical University Munich

[huseyin.coskun@tum.de](mailto:huseyin.coskun@tum.de)

Francesco Ferroni

Autonomous Intelligent Driving GmbH

[francesco.ferroni@aid-driving.eu](mailto:francesco.ferroni@aid-driving.eu)

Nassir Navab

Technical University Munich

[nassir.navab@tum.de](mailto:nassir.navab@tum.de)

Federico Tombari

Technical University Munich

Google

[tombari@in.tum.de](mailto:tombari@in.tum.de)

## Abstract

We present a sampling-free approach for computing the epistemic uncertainty of a neural network. Epistemic uncertainty is an important quantity for the deployment of deep neural networks in safety-critical applications, since it represents how much one can trust predictions on new data. Recently promising works were proposed using noise injection combined with Monte-Carlo sampling at inference time to estimate this quantity (e.g. Monte-Carlo dropout). Our main contribution is an approximation of the epistemic uncertainty estimated by these methods that does not require sampling, thus notably reducing the computational overhead. We apply our approach to large-scale visual tasks (i.e., semantic segmentation and depth regression) to demonstrate the advantages of our method compared to sampling-based approaches in terms of quality of the uncertainty estimates as well as of computational overhead.

## 1. Introduction

Quantifying the uncertainty associated with the prediction of neural networks is a prerequisite for their deployment and use in safety-critical applications. Whether used to detect road users and make driving decisions in an autonomous vehicle, or in a medical setting within a surgical robot, neural networks must be able not just to predict accurately, but also to quantify how certain they are regarding such predictions. Moreover, it is important that such uncertainty is provided during inference in real-time, so that the uncertainty can be exploited by the real-time safety-critical system.

One can estimate two types of uncertainty of a machine

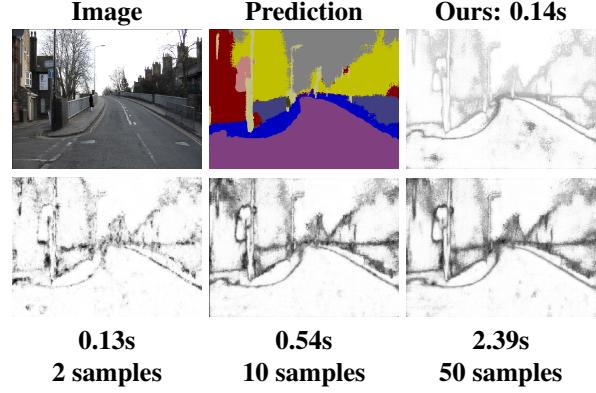


Figure 1. Qualitative results of uncertainty estimation for Bayesian SegNet (white: small, black: large). Upper: Original image (left), prediction using Monte-Carlo (MC) sampling (middle) and proposed uncertainty prediction (right, with average runtime per prediction). Lower: Uncertainty using MC dropout [9] using 2 (left), 10 (middle) and 50 (right) samples, each with its average runtime per prediction. We cache results prior to the first dropout layer to optimize performance of MC dropout.

learning model [8]: aleatoric and epistemic. Aleatoric uncertainty is inherent to the data itself, e.g. uncertainty resulting from noisy sensors. This type of uncertainty can be incorporated into the deep model itself by applying e.g. mixture density networks [2]. Epistemic uncertainty is the uncertainty in the chosen model parameters. In order to detect situations which are unfamiliar for a given machine learning model, and consequently quantify how much one can trust predictions on the given data, one has to determine the latter of the two types of uncertainty.

Recently many approaches have been proposed that make it possible to estimate epistemic uncertainty for large

scale neural network architectures[9, 26, 19, 22]. A promising research direction is the use of noise injection [9, 26] via, *e.g.*, stochastic regularization techniques. The underlying idea is to train a neural network while injecting noise at certain layers. During training, the network learns how to compensate the noise on the training data distribution, and thus minimize the variance of the prediction. At inference time, one can then use the variance in the prediction generated by different noise samples as an epistemic uncertainty estimate, since the neural network only learned to compensate the noise on the training data distribution.

Unfortunately, while these methods are conceptually simple and have been successful in delivering a measure of epistemic uncertainty for neural networks (even large architectures, *e.g.* [18, 10, 1]), they rely on Monte-Carlo (MC) sampling at inference time in order to determine the variance of the prediction as an uncertainty estimate. This means that computation time scales linearly with the number of samples, and therefore, can become prohibitively expensive for performance-critical or compute-limited applications, such as autonomous vehicles, robots and mobile devices. In such cases, obtaining epistemic uncertainty as part of a model prediction can be a functional safety requirement, but the need to perform real-time inference from sensor data makes MC dropout a difficult proposition.

In this work, we side-step these issues and produce epistemic uncertainty estimates of a neural network’s prediction which are at the same time accurate and computationally inexpensive. Our contributions are specifically:

- We propose a sampling-free approach to generally approximate uncertainty estimates of methods that rely on noise injection at training time.
- We simplify this approach specifically for the common case of convolutional neural networks combined with rectified linear units as activation function.

In the following sections we will firstly outline relevant work, secondly present our sampling-free framework and finally show experimental results. Specifically, we compare the quality of our approximation to Bayesian SegNet [18] on CamVid dataset [4], and show the ability of our approximation to detect out-of-distribution samples by training Bayesian SegNet while withholding certain classes at training time. We further apply our approximation in a common regression task for computer vision, *i.e.* monocular depth estimation [12].

## 2. Related Work

Recently there has been a wealth of proposals regarding epistemic uncertainty estimation for large-scale neural networks [3, 14, 22, 19, 9, 26, 29, 21, 23]. The common goal is

to approximate the full posterior distribution of the parameters of a neural network.

Some works aim to directly learn the parameters of a family of distributions within back-propagation [29, 3, 14]. Another line of research approximates the posterior by training ensembles of neural networks via random changes in the training setup [22, 19] estimating the target distribution by an ensemble of sample distributions [6].

A different research avenue utilizes stochastic regularization methods to estimate epistemic uncertainty at inference time [9, 26, 29]. The most prominent example is MC dropout [9] - train a dropout regularized neural network, and then, at inference time, keep dropout turned on to estimate the epistemic uncertainty via the variance of the prediction. These approaches have gained popularity due to the simplicity with which they integrate into the current training methodology. Consequently they have been applied to a variety of tasks [10, 1, 18, 7, 17]. Despite their achievements, they still suffer from large computational overhead at inference time due to sampling, which makes them prohibitively expensive in applications that demand real-time inference from large neural networks.

[15] optimizes the application of MC dropout on videos. Therefore the authors treat images which are close in time as constant, and thus samples of the same scene. Consequently each image only has to be processed once while performing approximate MC sampling.

Sampling-free estimation of epistemic uncertainty has been only partially covered in literature. [5] incorporates sampling-free epistemic uncertainty into the framework of mixture density networks which, following [20], suffers from non-convergence for high dimensional problems. Natural Parameter Networks [27, 16] can be considered related to our work. Instead of processing point estimates through a neural network, the authors adjust the transformations at each layer to propagate the natural parameters of a pre-defined distribution, *e.g.* mean and standard deviation of a Gaussian. Notably, these networks are limited in their choice of transformations as they are constrained to preserve the shape of the underlying distribution, and consequently do not generalize naturally to arbitrary large scale architectures. Thus we will not compare against this work since we are interested in approaches applicable to arbitrary neural networks without altering the training process.

## 3. Method

Our goal is to estimate the epistemic uncertainty of a neural network trained with injected noise at inference time to quantify the level of trust in the predictions, in a single shot. At its core our method uses error propagation [25], commonly used in physics, where the error is equivalent to the variance. We treat the noise injected in a neural network as errors on the activation values. By training with noise in-

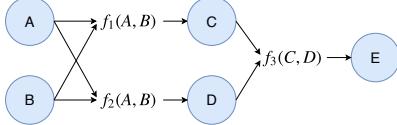


Figure 2. Simple computational graph for illustrating error propagation.

jection, the network implicitly learns to minimize the accumulated errors on the training data distribution, since larger errors correspond to large loss signals. To give an intuition for this, we consider a simple computational graph (see Fig. 2). Let  $A$  and  $B$  be independent random variables and let  $C = f_1(A, B)$  and  $D = f_2(A, B)$  be, possibly non-linear, functions of  $A$  and  $B$ . Knowing the mean and the variance  $\sigma_{A/B}^2$  of  $A$  and  $B$ , we want to compute the variance of  $C$  and  $D$ . We apply error propagation, where:

$$\sigma_{C/D}^2 = \left( \frac{\partial f_{1/2}}{\partial A} \right)^2 \sigma_A^2 + \left( \frac{\partial f_{1/2}}{\partial B} \right)^2 \sigma_B^2 \quad (1)$$

Note that the partial derivatives only approximate the outcome given non-linear functions  $f_{1/2}$ . Let us assume now that we have another function  $E = f_3(C, D)$ . We cannot apply Eq. 1 directly to determine the variance of  $E$  because  $C$  and  $D$ , unlike  $A$  and  $B$ , are not statistically independent. Thus we have to consider the full covariance matrix of  $C$  and  $D$  which we can obtain again by applying error propagation. We start with the covariance matrix  $\Sigma_{A,B}$  of  $A$  and  $B$ . This is obviously a diagonal matrix with entries  $\sigma_A^2$  and  $\sigma_B^2$  on its main diagonal. Then we can approximate the covariance matrix over  $C$  and  $D$  by computing:

$$\Sigma_{C,D} = J^T \Sigma_{A,B} J \quad (2)$$

Here  $J$  is the Jacobian of the vector-valued function  $\vec{f} = (f_1(A, B), f_2(A, B))^T$ . The final variance  $\sigma_E^2$  of  $E$  is obtained by applying Eq. 2 again but considering that  $f_3$  is not a vector-valued function. Thus one can exchange the Jacobian with the gradient  $J = \nabla_{C,D} f_3(C, D) = (\frac{\partial}{\partial C}, \frac{\partial}{\partial D})^T f_3(C, D)$ .

This minimal example already illustrates all the tools we need to approximate the variance at the output layer of a neural network given some noise layer, such as dropout or batch-norm [26] by applying error propagation. In the following, we explain the propagation of covariance for specific parts of a neural network: noise layers, affine layers, and non-linearities. Afterwards, we simplify the above equations for the common setup of a convolution and ReLU activation. This is necessary for high dimensional feature spaces due to the size of the covariance matrix (which scales quadratically with the number of activations).

We explicitly do not model the impact of correlations between activations on the activation means as we explore the

applicability of performing variance propagation using error propagation in neural networks and leave it to future work to incorporate it into the computational framework.

### 3.1. Noise Layer

Now we derive the covariance matrix of a noise layer's activation values in a neural network. The following consideration is independent of the implementation of the noise layer. We assume independent noise across the nodes of the noise layer which is not a necessity but common practice. In the following, superscripts correspond to layers and subscripts to nodes within a layer.

Consider a neural network with  $l$  layers and  $N$  noise layers at positions  $i \in [0, l]$ , where  $l = 0$  denotes the input. Let the input to a given noise layer be described by a random vector  $\vec{X}$  with covariance matrix  $\Sigma^{i-1}$ . Furthermore, let the random vector representing the noise be  $\vec{Z} \in \mathbb{R}^n$  with a diagonal covariance matrix  $\Sigma_{\vec{Z}}$ , where the entries of  $\vec{Z}$  are independent and the entries of  $\vec{X}$  are generally *dependent*. There are two ways how the noise is commonly injected: addition and element-wise multiplication of  $\vec{X}$  and  $\vec{Z}$ .

When the noise injection is represented by adding the random vectors the resulting covariance at layer  $i$  is simply given by

$$\Sigma^i = \Sigma^{i-1} + \Sigma_{\vec{Z}} \quad (3)$$

When the noise injection resembles an element-wise multiplication of the random vectors  $\vec{X}$  and  $\vec{Z}$  (e.g. dropout), the covariance matrix at layer  $i$  is given by [13]

$$\begin{aligned} \Sigma^i &= \Sigma_{\vec{Z} \circ \vec{X}, \vec{Z} \circ \vec{X}} = \\ &\Sigma_{\vec{Z}} \circ \Sigma^{i-1} + E[\vec{Z}] E[\vec{Z}]^T \circ \Sigma^{i-1} + E[\vec{X}] E[\vec{X}]^T \circ \Sigma_{\vec{Z}} \end{aligned} \quad (4)$$

where  $\circ$  is the Hadamard product. We refer to the supplementary material for a detailed derivation of this formula.

For the special case of the first noise layer in the network, we can either model the input noise from prior knowledge (*i.e.* sensor noise) or simplify it by assuming zero noise. In the latter case, the resulting covariance matrix will be diagonal given independent noise, resulting in:

$$\Sigma^i = \text{diag}(\sigma_0^2, \dots, \sigma_n^2) \quad (5)$$

where  $n$  is the dimensionality of the activation vector and  $\sigma_i^2$  is the variance of activation  $i$ . The variance introduced by the regular dropout, which follows a Bernoulli distribution, is given by  $p(1-p)a_i^2$ , with  $p$  defining the dropout rate and  $a_i$  the mean activation of node  $i$ .

### 3.2. Affine Layers and Non-Linearities

After obtaining the covariance matrix of a noise layer one needs to propagate it to the output layer. This usually means applying a series of affine layers and non-linearities.

Here, we detail the case of a fully-connected and convolutional layer. It is straightforward to apply Eq. 2, given that for the transformation of an affine layer the Jacobian  $J$  equals the weight matrix  $W$ . The covariance matrix is therefore,

$$\Sigma^i = W \Sigma^{i-1} W^T \quad (6)$$

This is an exact transformation which does not depend on the underlying distribution. For the non-linearities in a neural network we approximate the transformation by a first-order Taylor expansion. The covariance transformation at a non-linearity is then given by:

$$\Sigma^i \approx J \Sigma^{i-1} J^T \quad (7)$$

The particular Jacobians of the activation functions used in our experiments (ReLU, sigmoid and softmax) can be found in the supplementary material where we also provide an analysis of the error introduced by the first-order Taylor expansion of the softmax activation function.

### 3.3. Special Case: Convolutional Layers combined with ReLU Activations

Though the proposed approach does not require sampling, propagating the full covariance matrix may become prohibitively expensive for very high dimensional problems, such as images. This can be understood by considering that our method requires the full covariance matrix  $\Sigma \in \mathbb{R}^{N \times N}$  at each layer with  $N$  nodes. This leads to a memory complexity of  $O(N^2)$ . Since the many neural network architecture contains iterative applications of convolutional layers and rectified linear units, we simplify the above formulas for the sake of computational efficiency.

For a fully connected layer, since each of its output nodes is a linear combination of all input nodes, modeling the full covariance is a necessity. However, this is not the case for a convolutional layer which strength comes from sharing weights across the entire input space and consequently applying a linear transformation only to a local neighborhood of each pixel. For the following approximation we assume a convolutional layer with kernel  $K \in \mathbb{R}^{W' \times H' \times C'}$  and an input to the convolutional layer  $I \in \mathbb{R}^{W \times H \times C}$  with  $W' \ll W$  and  $H' \ll H$ . Given a diagonal covariance matrix prior to the convolutional layer, by applying e.g. a dropout layer, the output covariance would be a sparse matrix with a few non-zero entries off the main diagonal for local neighbourhoods in the input space. This is illustrated in Fig. 3 using a convolutional architecture on CIFAR10. Furthermore, given approximately symmetrical distributed weights ReLU activation leads to a probability of roughly 0.5 with which a variance value is dropped. This results in the observed decrease in the mean variance with the number of convolutional layers using ReLU activation functions.

The observation that this architectural setup does not transport significant mass to wide regions of the covariance matrix motivates us to assume a *diagonal covariance matrix*.

As a result the computational complexity of propagating the variance reduces to the same level of normal forward propagation because one only needs to propagate the vector of the main diagonal of the covariance matrix. Under this assumption Eq. 4 simplifies to:

$$V[\vec{X}^i] = E[\vec{X}^{i-1}]^2 \circ V[\vec{Z}] + E[\vec{Z}]^2 \circ V[X^{i-1}] + V[\vec{X}^{i-1}] \circ V[\vec{Z}] \quad (8)$$

A derivation can be found in the supplementary material. To determine variances under the transformation of weight matrices and Jacobians of non-linearities using the above simplification, we need to square the corresponding matrix element-wise and multiply it with the vector representing the main diagonal of the covariance matrix:

$$Var[\vec{X}^i] = (W^2)Var[\vec{X}^{i-1}] \quad (9)$$

and respectively

$$Var[\vec{X}^i] \approx (J^2)Var[\vec{X}^{i-1}] \quad (10)$$

With the assumption of independent activations it is now straightforward to improve upon the simple Jacobian approximation of ReLU by explicitly computing the variance resulting from applying ReLU to a Gaussian. Therefore we assume a Gaussian distribution of activations prior to ReLU which is valid according to [28]. The respective formulas and derivations can be found in the supplementary material.

## 4. Experiments

In this section we provide experimental evidence that our approximation can produce fast and accurate uncertainty estimates in a classification and regression setting.

### 4.1. Synthetic Data

First we want to give an intuition for the validity of our approach by applying it to a synthetic dataset, and comparing it directly to MC sampling. We create a regression dataset that consists of a single input and a single output, where the input is uniformly distributed in the interval  $[0, 20]$  and the target is the sine of the input plus Gaussian noise of mean  $\mu = 0$  and sigma  $\sigma = 0.3$ . We fit a fully connected neural network with three hidden layers each containing 100 hidden units to the data. We apply dropout ( $p = 0.1$ ) prior to the last hidden layer. Since our primary goal is to approximate the epistemic uncertainty, we populate the test set with out-of-distribution samples (i.e. samples smaller than 0 or larger than 20) alongside samples from the training data distribution  $[0, 20]$ . We approximate the variance, and thus the standard deviation, of the

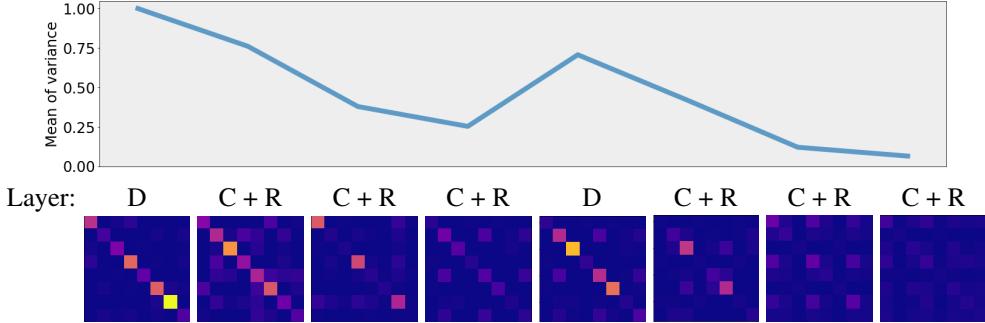


Figure 3. We train a neural network (convolution - DropoutBlock - fully connected layer, where DropoutBlock corresponds to above sequence of dropout (D), convolution (C) and ReLU (R)) on CIFAR10. The feature maps in the DropoutBlock have the dimensionality  $10 \times 10 \times 3$  and the convolutional filters are of dimensionality  $3 \times 3 \times 3$ . The upper part shows the mean variance of the activation values normalized with respect to the first dropout layer (blue). The lower part shows images (zoomed in for better visualization) of the mean covariance matrix at the corresponding layer (blue: small absolute values, yellow: large absolute value). We make two observations. Firstly, dropout strengthens the main diagonal. Secondly, variance decays in absence of additional dropout layers. Consequently, large regions of the covariance matrix stay approximately zero.

prediction in two ways - MC sampling (100 samples) and propagating the variance with our approximation.

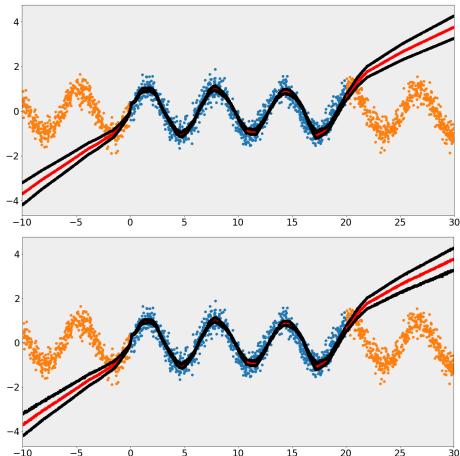


Figure 4. Synthetic data. The neural network has three hidden layers with each 100 hidden units. We drop units prior to the last layer with probability  $p = 0.1$ . We plot samples from the training data distribution (blue), samples outside of the training data distribution (orange), the prediction (red) and the prediction plus/minus the standard deviation (black). Upper: The standard deviation is approximated following our approach. Lower: The standard deviation is determined using MC dropout [9] with 100 samples.

Fig. 4 visualizes the results for both approaches. Characteristically for epistemic uncertainty, we observe that the standard deviation of the prediction outside of the training data distribution increases. For this example, our approximation is in fact exact. In the supplementary material we show empirically that the sampling-based variance estimate converges towards our analytic estimate for large number of samples.

## 4.2. Predictive Performance

Following prior work estimating epistemic uncertainty [14, 9, 26] we analyze the predictive performance on 9 of the original 10 UCI regression datasets. We leave out the Year Prediction MSD dataset as done by other recent work [26]. We only compare our approach directly to MC dropout since we approximate the latter.

We evaluate two metrics - root mean squared error (RMSE) on the test set and test log-likelihood (TLL). We expect our RMSE to be higher than MC dropout, as dropout sampling is a better approximation than scaling activations [9]. The TLL represents the value of interest in our experiments as it quantifies the quality of the predicted distribution. It measures the probability mass on the target without making assumptions on the underlying distribution.

We follow the original setup in [14]<sup>1</sup>. We split the training data 20 times randomly into training and validation set, except for the dataset Protein Structure where we use five splits, and perform a separate grid search for the hyperparameters dropout rate and  $\tau$ . Following [14, 9] we use one hidden layer with 50 hidden units, except for Protein Structure where we use 100 hidden units. Dropout is applied directly to the input and after the hidden layer and we train the network for 400 epochs. We use the full covariance matrix to propagate uncertainty, defined in Eq 4, 6, 7.

The TLL in the original experiment requires sampling from distributions of outputs. Since our method naturally just returns the parameters of a unimodal distribution over the outputs, we assume a Gaussian distribution and sample from it to compute the TLL. According to [28] this is a reasonable assumption given our hidden layer dimension. We

<sup>1</sup>Instead of using Bayesian optimization [24] for hyper-parameter optimization we use grid search

Dataset	Test RMSE		Test log-likelihood		Runtime [s]	
	MC[9]	OUR	MC[9]	OUR	MC[9]	OUR
Boston Housing	<b>3.06 ± 0.18</b>	3.13 ± 0.22	<b>-2.55 ± 0.07</b>	-2.65 ± 0.12	3.47	<b>0.06</b>
Concrete Strength	<b>5.42 ± 0.10</b>	<b>5.42 ± 0.11</b>	<b>-3.11 ± 0.02</b>	-3.13 ± 0.02	3.63	<b>0.06</b>
Energy Efficiency	1.60 ± 0.05	<b>1.59 ± 0.05</b>	<b>-1.91 ± 0.03</b>	-1.96 ± 0.03	3.27	<b>0.06</b>
Kin8nm	<b>0.08 ± 0.00</b>	<b>0.08 ± 0.00</b>	1.10 ± 0.01	<b>1.11 ± 0.01</b>	4.75	<b>0.06</b>
Naval Propulsion	<b>0.00 ± 0.00</b>	<b>0.00 ± 0.00</b>	<b>4.36 ± 0.01</b>	3.64 ± 0.02	5.10	<b>0.06</b>
Power Plant	<b>4.04 ± 0.04</b>	4.05 ± 0.04	<b>-2.82 ± 0.01</b>	-2.85 ± 0.01	4.46	<b>0.06</b>
Protein Structure	<b>4.42 ± 0.03</b>	<b>4.42 ± 0.03</b>	<b>-2.90 ± 0.01</b>	<b>-2.90 ± 0.00</b>	4.38	<b>0.06</b>
Wine Quality Red	<b>0.63 ± 0.01</b>	<b>0.63 ± 0.01</b>	<b>-0.95 ± 0.02</b>	<b>-0.95 ± 0.01</b>	3.49	<b>0.06</b>
Yacht Hydrodynamics	<b>2.89 ± 0.25</b>	3.14 ± 0.31	-2.32 ± 0.10	<b>-2.10 ± 0.07</b>	3.42	<b>0.06</b>

Table 1. table

RMSE, test log-likelihood and runtime for MC dropout (MC) [9] and our approximation (OUR). The error, denoted by  $\pm$ , is the standard error. For the RMSE smaller values are better, for the TLL larger values. We follow the original setup in [9] and use  $T=10000$  samples with MC dropout. Our runtime is 0.06 in every row due to decimal precision.

perform the same grid search as for MC dropout. For both, MC dropout and our proposed approximation, we sample 10000 predictions to compute the TLL.

Table 1 shows the results of this experiment. The TLL of our approximation is almost for every regression dataset only marginally lower or even larger than the TLL obtained by MC dropout. It is only for the Naval Propulsion dataset much worse than the original sampling-based method. However, given a RMSE of  $0.00 \pm 0.00$  we perfectly fit this dataset. In this case, extremely confident and accurate predictions may lead to a regime where the Gaussian assumption loses validity or higher accuracy of MC sampling has a stronger impact on the TLL. Given the deviation of other methods from MC dropout, which can be found in [9], our approximation performs well.

### 4.3. Classification Task: Bayesian SegNet

A recent large scale architecture applying MC dropout is Bayesian SegNet [18] for semantic segmentation. The original work examines several architectures differing in the placement of dropout within the network. According to the authors, for the quality of the uncertainty estimate the location of dropout within the architecture is irrelevant. We train consider Bayesian SegNet on CamVid dataset [4] and investigate two of their probabilistic variants:

- Dropout after the central four encoder and decoder blocks (ENCDEC). This is the best performing setup in terms of prediction [18]. It is challenging for our approximation since the last dropout layer is placed far away from the output layer. Thus our methods naturally underestimates the variance of the prediction since we assume a diagonal covariance matrix.
- One dropout layer before the final classifier layer (CLASS). This setup fits better our method, since the

Method	G	C	I/U
Original Bayesian SegNet[18]	<b>86.9</b>	76.3	<b>63.1</b>
Our implementation	86.1	<b>76.4</b>	54.1

Table 2. table

We compare our retrained Bayesian SegNet (ENCDEC architecture) [18] to the results presented in the original work. We compare the quantities global accuracy (G), average class accuracy (C) and mean intersection over union.

dropout layer is closer to the output, this reducing the number of error sources.

We compare the performance of our implementation to the original work in Table 2. We emphasize that the goal of this work is not to outperform the state of art, but rather to provide an efficient sampling-free approximation of epistemic uncertainty. We thus aim at showcasing the similarity between sampling-based uncertainty estimation and the proposed approximation.

We trained with batch size 8 and stochastic gradient descent with initial learning rate 0.1 and exponential learning rate decay with base 0.95 for 200 epochs. We use early stopping (watching the validation loss) with patience 50. The original images in CamVid have resolution 720x960 and 32 classes. Following [18] we only use 11 generalized classes and downsample the images to 360x480.

Since semantic segmentation is a classification task, one obtains variances for each class at each pixel. There exist several strategies to aggregate these variance into a scalar quantity (for a selective list see [10]). Subsequently we follow the original work [18] and use the mean standard deviation of the softmax scores at each pixel. We apply our approximation according to Eq. 8, 9, 10.

Fig. 5 and 1 show qualitative results of our approxima-

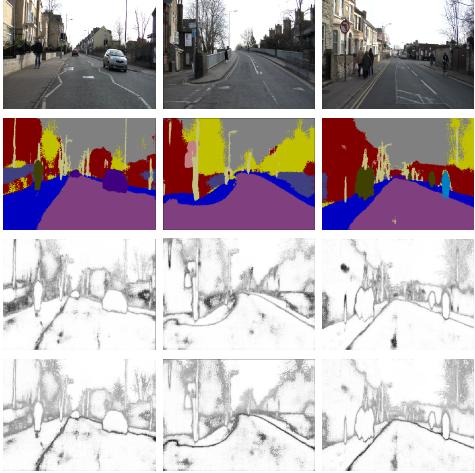


Figure 5. Qualitative results of our approximation. First row: Input image. Second row: Segmentation result using MC dropout [9] with 50 samples. Third row: Uncertainty estimate using MC dropout. Fourth row: Our approximation. We use differently scaled colormaps for the uncertainty maps to emphasize their structural similarity.

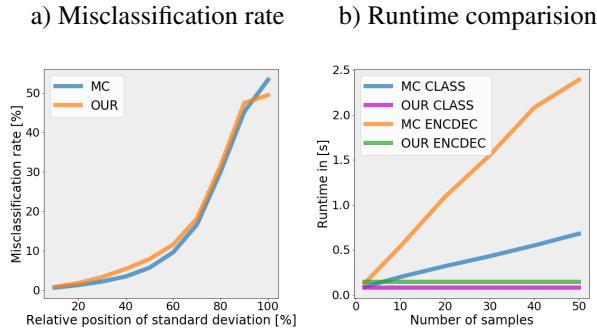


Figure 6. a): Comparison of pixel misclassification rate depending on quantile of standard deviation between MC dropout [9] with 50 samples (blue, MC) and our approximation (orange, OUR) (e.g. 50% implies that 50% of the pixels have a smaller standard deviation.) b): Runtime comparison between MC dropout and our approximation (OUR) for both architectures (ENCDEC and CLASS). Our approximation (ENCDEC: green, CLASS: red) is constant (no sampling) and MC dropout (ENCDEC: orange, CLASS: blue) increases linearly with the number of samples. We cache results prior to the first dropout layer to optimize performance of MC dropout.

tion and the sampling-based estimates (a video can be found in the supplementary material). We first approximate the uncertainty estimates using the ENCDEC architecture. For all qualitative results we only show the image, MC prediction, MC uncertainty estimate and our approximation, since our work focuses on producing sampling-free uncertainty estimates. For qualitative examples including our prediction and the ground truth we refer to the supplementary material. Both methods predict similar regions of high uncertainty.

Those are mainly the object boundaries resulting from the noise induced by human labelers. Given the qualitative similarity of the predicted uncertainty, we surprisingly found that the magnitude of the approximated uncertainty is much lower than the sampling-based uncertainty (mean absolute difference is 93.7% of the mean variance received by MC sampling). We understand that the difference arises from the fact that in this architecture the last dropout layer is far away from the output layer. Given the qualitative similarity of the predicted uncertainty, we deduce that mixture terms, which lie off the main diagonal of the covariance matrix, primarily act as a variance bias.

Fig. 6 compares the runtime of sampling-based uncertainty estimation with our approximation. We cache the results prior to the first dropout layer in the architecture and only repeatedly propagate the part of the network that contains dropout layers to optimize MC sampling. As expected we observe a linear dependence on the number of samples of the sampling-based approach where the slope depends on the location of the first dropout layer. We clearly see the computational advantage of our approximation.

To prove coherence of the predicted uncertainties on the training data distribution, we investigate the correlation of the uncertainty with the misclassification rate on the test set. Since we are explicitly interested in epistemic uncertainty, we further prove that our uncertainty estimate increases for out-of-distribution samples.

We plot the pixel misclassification rate against the uncertainty value (see Fig. 6). Obviously we wish to observe an increasing pixel misclassification rate with larger uncertainty estimates. Since the scale of magnitude of our uncertainty estimates is different from the sampling-based approach we do not plot it directly against the uncertainty value. To be able to compare their behavior directly, we plot the misclassification rate against quantile of the uncertainty estimate. Both curves behave similarly which implies similar calibration of the uncertainty.

Moreover, we use the CLASS architecture to investigate whether our approximation is able to detect out-of-distribution samples. It is difficult to design adequate experiments to validate this capability of an epistemic uncertainty estimate. Here, we withhold certain classes during training time and present them to the network at test time. This way we exclude respective regions of the training data distribution. We exclude two classes: pedestrians and cyclists. We selected them based on the fact that they differ from other classes in their appearance, are similar to each other and do not make up large enough areas of the images to endanger convergence of the training. Qualitative results of this training can be found in the supplementary material. Fig 7 shows that the mean uncertainty value of each classes with and without withholding classes. The average relative increase of uncertainty is maximal for withheld classes.

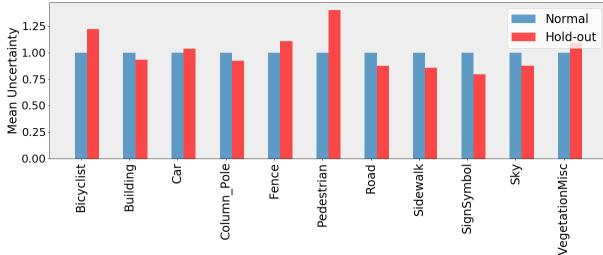


Figure 7. Mean uncertainty (using our approximation) per class for CLASS architecture using all classes (blue) and withholding the classes for pedestrians and cyclists (red). Uncertainties are normalized to the uncertainty value observed using all classes to highlight their relative change. The absolute value strongly depends on the boundary to area ratio of each class since high uncertainties mostly occur at boundaries.

#### 4.4. Regression Task: Depth Regression

We now evaluate our approach on a regression task. Without a softmax, we expect our approximation to not only show similar behavior but also to be compellingly close to the sampling-based approach. We apply our approximation to monocular depth regression [12]. The final activation of this architecture is a sigmoid function. Thus it is expected that our approximation will not exactly match the sampling-based result. Following the original work we train on the KITTI dataset [11] and keep the setup unchanged with exception of inserting a dropout layer prior to the final convolution to estimate uncertainty. As uncertainty estimate we choose the variance of the regression output and we use Eq. 8, 9 and 10 to propagate variance. Note that this setup does not require modeling the full covariance matrix because sigmoid is an element-wise operation.

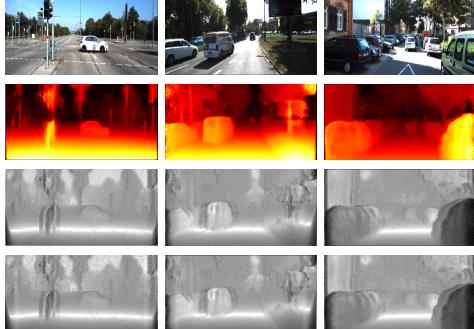


Figure 8. Qualitative results of our depth regression network. First row: Input image. Second row: Depth regression using MC dropout [9]. Third row: Logarithm of variance (white: low, black: high) using MC dropout with 50 samples. We use the logarithm because high uncertainties from non-overlapping stereo images on the left and right of the image are dominating the colormap. Fourth row: logarithm of variance using our approximation.

Fig. 8 shows qualitative results of this experiment. We visualize the logarithm of the uncertainty value since high

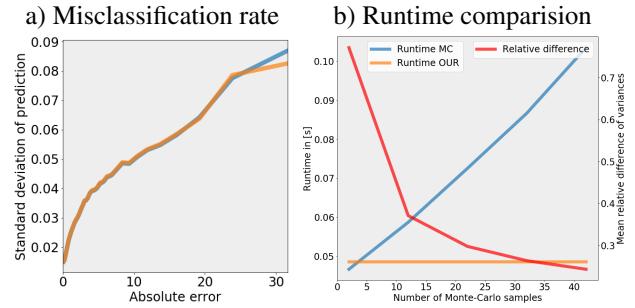


Figure 9. a): Standard deviation of depth prediction depending on the absolute difference between the predicted depth and the ground truth. The standard deviation for both, MC dropout [9] with 50 samples (blue) and our approximation (orange), behave similarly. b): Runtime of our method (orange) and MC dropout [9] with 50 samples (blue) depending on the number of samples. We visualize the mean relative difference between our variance approximation and the variance computation based on MC dropout (red).

uncertainty at the left and the right border is dominating the colormap. This is an artifact of the underlying training method. Given stereo pairs of images the network learns to predict disparity maps transforming one image into the other. Due to non-overlapping borders, reconstruction fails in those regions and consequently the uncertainty is high. Furthermore we consistently observe an uncertainty minimum at a certain depth. This is the area of optimal depth resolution of the stereo camera setup. The depth resolution of a point in space increases quadratically with its distance from the camera. On the contrary the depth resolution is proportional to the disparity error which increases for closer objects due to larger disparities.

We analyze how the mean absolute variance difference relative to the mean variance and runtime depend on the number of MC samples. The result is displayed in Fig. 9 b). As expected the runtime of the sampling-based approach increases linearly with the number of samples. On the other hand the mean absolute difference decreases with the number of samples. This suggests that our approach is not only superior in terms of runtime but also in terms of the uncertainty estimate. The fact that the relative difference does not converge to zero originates from the usage of the sigmoid activation function after the final layer.

Finally we show the meaningfulness of our approximated uncertainty predictions and the sampling-based results by plotting their correlation with the absolute difference between the predicted depth and the ground truth (see Fig. 9 a)). The standard deviation of both, the sampling-based result and our approximation of it, increase linearly with the absolute error. Thus our uncertainty estimate can be used to identify regions which are likely to have large regression errors while adding minimal computational overhead.

## 5. Conclusion

We have shown that the framework of error propagation can be used to approximate the sampling procedure of epistemic uncertainty estimates that rely on noise injection at training time. We applied the proposed approximation to two large scale computer vision tasks illustrating the computational efficiency and the coherence of the resulting uncertainty maps. The approximation is numerically better for noise layer located closer to the output layer. Having a methodology to analytically approximate the uncertainty estimate based on stochastic regularization, in future research we aim to represent the noise injection by a loss function which will enable us to learn the noise parameter (e.g. dropout rate). This has the potential to provide an estimate for *aleatoric and epistemic* uncertainty.

## 6. Acknowledgements

We thank Nutan Chen and Seong Tae Kim for the valuable discussions and constructive feedback. We further thank Autonomous Intelligent Driving GmbH for supporting this work.

## References

- [1] A. Bhattacharyya, M. Fritz, and B. Schiele. Long-term on-board prediction of people in traffic scenes under uncertainty. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4194–4202, 2018.
- [2] C. M. Bishop. Mixture density networks. Technical report, Citeseer, 1994.
- [3] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622, 2015.
- [4] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.
- [5] S. Choi, K. Lee, S. Lim, and S. Oh. Uncertainty-aware learning from demonstration using mixture density networks with sampling-free variance modeling. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6915–6922. IEEE, 2018.
- [6] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.
- [7] 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)*, pages 3266–3273. IEEE, 2018.
- [8] Y. Gal. *Uncertainty in deep learning*. PhD thesis, PhD thesis, University of Cambridge, 2016.
- [9] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.
- [10] Y. Gal, R. Islam, and Z. Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1183–1192. JMLR.org, 2017.
- [11] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.
- [12] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.
- [13] L. A. Goodman. On the exact variance of products. *Journal of the American statistical association*, 55(292):708–713, 1960.
- [14] J. M. Hernández-Lobato and R. Adams. Probabilistic back-propagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.
- [15] P.-Y. Huang, W.-T. Hsu, C.-Y. Chiu, T.-F. Wu, and M. Sun. Efficient uncertainty estimation for semantic segmentation in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 520–535, 2018.
- [16] S. J. Hwang, R. Mehta, and V. Singh. Sampling-free uncertainty estimation in gated recurrent units with exponential families. *arXiv preprint arXiv:1804.07351*, 2018.
- [17] M. Kampffmeyer, A.-B. Salberg, and R. Jenssen. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–9, 2016.
- [18] A. Kendall, V. Badrinarayanan, and R. Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *CoRR*, abs/1511.02680, 2015.
- [19] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.
- [20] 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)*, pages 3873–3878. IEEE, 2018.
- [21] M. Możejko, M. Susik, and R. Karczewski. Inhibited softmax for uncertainty estimation in neural networks. *arXiv preprint arXiv:1810.01861*, 2018.
- [22] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016.
- [23] C. Rupprecht, I. Laina, R. DiPietro, M. Baust, F. Tombari, N. Navab, and G. D. Hager. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3591–3600, 2017.
- [24] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances*

- in neural information processing systems*, pages 2951–2959, 2012.
- [25] J. Taylor. *Introduction to error analysis, the study of uncertainties in physical measurements*. 1997.
  - [26] M. Teye, H. Azizpour, and K. Smith. Bayesian uncertainty estimation for batch normalized deep networks. In *International Conference on Machine Learning*, pages 4914–4923, 2018.
  - [27] H. Wang, S. Xingjian, and D.-Y. Yeung. Natural-parameter networks: A class of probabilistic neural networks. In *Advances in Neural Information Processing Systems*, pages 118–126, 2016.
  - [28] S. Wang and C. Manning. Fast dropout training. In *international conference on machine learning*, pages 118–126, 2013.
  - [29] G. Zhang, S. Sun, D. Duvenaud, and R. Grosse. Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pages 5847–5856, 2018.