Correlated Parameters to Accurately Measure Uncertainty in Deep Neural Networks

Konstantin Posch and Juergen Pilz

Abstract—**In this article, a novel approach for training deep neural networks using Bayesian techniques is presented. The Bayesian methodology allows for an easy evaluation of model uncertainty and, additionally, is robust to overfitting. These are commonly the two main problems classical, i.e., non-Bayesian architectures have to struggle with. The proposed approach applies variational inference in order to approximate the intractable posterior distribution. In particular, the variational distribution is defined as the product of multiple multivariate normal distributions with tridiagonal covariance matrices. Every single normal distribution belongs either to the weights or to the biases corresponding to one network layer. The layerwise *a posteriori* variances are defined based on the corresponding expectation values, and furthermore, the correlations are assumed to be identical. Therefore, only a few additional parameters need to be optimized compared with non-Bayesian settings. The performance of the new approach is evaluated and compared with other recently developed Bayesian methods. Basis of the performance evaluations are the popular benchmark data sets MNIST and CIFAR-10. Among the considered approaches, the proposed one shows the best predictive accuracy. Moreover, extensive evaluations of the provided prediction uncertainty information indicate that the new approach often yields more useful uncertainty estimates than the comparison methods.**

Index Terms—**Bayesian statistics, convolutional neural networks (CNNs), deep learning, model uncertainty, parameter correlations, variational inference.**

I. Introduction

NOWADAYS, due to excellent results obtained in many fields of applied machine learning, including computer vision and natural language processing [1], the popularity of deep learning is increasing rapidly. One of the reasons can surely be found in the fact that Krizhevsky *et al.* [2] outperformed the competitors in the ImageNet Large Scale Visual Recognition Challenge 2012 by proposing a convolutional neural network (CNN) named AlexNet. While AlexNet includes eight layers, more recent architectures for image classification go even deeper [3], [4]. It is well known that a feedforward network with merely one hidden layer can approximate a broad class of functions arbitrarily well. A mathematical more profound formulation of this statement, i.e., the so-called universal approximation theorem, was proven by Hornik *et al.* [5]. However, Liang and Srikant [6] have proven that deep nets often require exponential fewer parameters than shallow ones in order to achieve a given degree of approximation. Possible applications of deep nets for computer vision include medical imaging, psychology, automotive industry, finance, and life sciences [7]–[12].

Deep learning has a large number of real-world use cases. Nevertheless, there are two restrictions that still limit its areas of application. The first restriction is that deep networks require a large amount of training data; otherwise, they are prone to overfitting. The reason for this is the huge amount of parameters neural nets hold. Although deep nets often require exponential fewer parameters than shallow ones in order to approximate a given function well, the remaining number is still very high. Thus, in many potential fields of application, where such an amount cannot be provided, deep learning is of limited use or often even cannot be used. To counteract this problem, commonly diverse regularization techniques are applied. Besides classical approaches, such as the penalization of the L2-norm or the L1-norm, stochastic regularization methods gain increasing attention. For instance, dropout [13] and dropconnect [14] count to these stochastic techniques. The first one randomly sets the activation of nonoutput neurons to zero during network training, and the second one randomly sets network weights to zero. While dropout is classically interpreted as an efficient way of performing model averaging with neural networks, Gal and Ghahramani [15] and Gal [16] as well as Kingma *et al.* [17] recently showed that it can also be considered as an application of Bayesian statistics. The second restriction classical deep networks struggle with is that model uncertainty cannot be measured. Considering that the model uncertainty can directly be translated to prediction uncertainty, this is a critical aspect for many applications. Especially, in the medical field or for self-driving vehicles, it is essential that the prediction uncertainty can be determined [18]. In these areas of application, a model that predicts, on average, quite well is not good enough. One has to know if the model is certain in its predictions or not, such that in the case of high uncertainty, a human can decide instead of the machine. In [19], a literature survey on neural network-based methods for uncertainty quantification using

Manuscript received April 2, 2019; revised November 6, 2019 and January 28, 2020; accepted March 5, 2020. Date of publication April 17, 2020; date of current version March 1, 2021. This work was supported by the ECSEL Joint Undertaking (JU) under Grant Agreement 783163; JU received support from the European Union's Horizon 2020 Research and Innovation Program. It is cofunded by the consortium members and grants from Austria, Germany, Belgium, Italy, Spain, and Romania. *(Corresponding author: Konstantin Posch.)*

The authors are with the Department of Statistics, University of Klagenfurt, 9020 Klagenfurt, Austria (e-mail: konstantin.posch@aau.at; juergen.pilz@aau.at).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2020.2980004

prediction intervals is given. Please be aware of the fact that the probabilities obtained when running a conventional deep net for a classification task should not be interpreted as the confidence of the model. As a matter of fact, a neural net can guess randomly while returning a high-class probability [20].

The consideration of uncertainties in models for real-life problems is receiving increasing attention and, by far, not only of interest in deep learning. For instance, in game theory, taking into account uncertainty is quite common. In this field, studies often use fuzzy and vague sets in their discussions of the game problems [21]–[23].

A possible strategy to overcome the restrictions classical deep learning has to deal with is applying Bayesian statistics. In so-called Bayesian deep learning, the network parameters are treated as random variables and are not considered to be fixed deterministic values. In particular, an *a priori* distribution is assigned to them, and updating the prior knowledge after observing training data results in the so-called posterior distribution. The uncertainty regarding the network parameters can be directly translated in uncertainty about predictions. Furthermore, Bayesian methods are robust to overfitting because of the built-in regularization due to the prior. It should be mentioned that besides the Bayesian methodology, there are also some other notable approaches to measure prediction uncertainty. In particular, ensemble learning and adversarial training gain increasing attention [24], [25].

Buntine and Weigend [26] were among the first who presented approximate Bayesian methods for neural nets. Two years later, Hinton and van Camp [27] proposed the first variational methods. Variational methods try to approximate the true posterior distribution with another parametric distribution, the so-called variational distribution. The approximation takes place due to an optimization of the parameters of the variational distribution. They followed the idea that there should be much less information in the weights than in the output vectors of the training cases in order to allow for a good generalization of neural networks. Denker and Lecun [28] as well as MacKay [29] used the Laplace approximation in order to investigate the posterior distributions of neural nets. Neal [30] proposed and investigated hybrid Monte Carlo training for neural networks as a less limited alternative to the Laplace approximation. However, the approaches mentioned up until now are often not scalable for modern applications that go along with highly parameterized networks. Graves [31] was the first to show how variational inference can be applied to modern deep neural networks due to the application of Monte Carlo integration. He used a Gaussian distribution with a diagonal covariance matrix as variational distribution. Blundell *et al.* [32] extended and improved the work of Graves [31] and also used a diagonal Gaussian to approximate the posterior. As mentioned earlier, Gal and Ghahramani [15] and Gal [16] as well as Kingma *et al.* [17] showed that using the regularization technique dropout can also be considered as variational inference.

The independence assumptions going along with variational inference via diagonal Gaussians (complete independence of network parameters) or also going along with variational inference according to dropout (independence within neurons) are restrictive. Permitting an exchange of information between different parts of neural network architectures should lead to more accurate uncertainty estimates. Louizos and Welling [33] used a distribution over random matrices in order to define the variational distribution. Thus, they could reduce the amount of variance-related parameters that have to be estimated and further allow for information sharing between the network weights. Note that in the diagonal Gaussian approach, assigning one variance term to each random weight and one variance term to each random bias doubles the number of parameters to optimize in comparison to frequentist deep learning. Consequently, network training becomes more complicated and computationally expensive [20].

It should be mentioned that variational Bayes is just a specific case of local α-divergence minimization. According to Amari [34], the α-divergence between two densities $p(\mathbf{w})$ and $q(\mathbf{w})$ is given by $D_\alpha(p(\mathbf{w})||q(\mathbf{w})) = 1/(\alpha(1-\alpha))\big(1-\int p(\mathbf{w})^\alpha q(\mathbf{w})^{(1-\alpha)}\, d\mathbf{w}\big)$. Thus, the α-divergence converges for $\alpha \to 0$ to the KL divergence [35], which is typically used in variational Bayes. It has been shown [36], [37] that an optimal choice of α is task specific and that nonstandard settings, i.e., settings with $\alpha \neq 0$ can lead to better prediction results and uncertainty estimates.

However, in this article, we do not propose an optimal choice of α. Rather, for the classical case $\alpha = 0$, we propose a good and easy to interpret variational distribution. For this task, recent work from Posch *et al.* [38] is extended. They used a product of Gaussian distributions with specific diagonal covariance matrices in order to define the variational distribution. In particular, the *a posteriori* uncertainty of the network parameters is represented per network layer and depending on the estimated parameter expectation values. Therefore, only a few additional parameters have to be optimized compared with classical deep learning, and the parameter uncertainty itself can easily be analyzed per network layer. We extend this distribution by allowing network parameters to be correlated with each other. In particular, the diagonal covariance matrices are replaced by tridiagonal ones. Each tridiagonal matrix is defined in such a way that the correlations between neighbored parameters are identical. This way of treating network layers as units in terms of dependence allows for easy analysis of the dependence between network parameters. Moreover, again only a few additional parameters compared with classical deep learning need to be optimized, which guarantees that the difficulty of the network optimization does not increase significantly. Note that our extension allows for an exchange of information between different parts of the network and, therefore, should lead to more reliable uncertainty estimates. We have evaluated our approach on the basis of the popular benchmark data sets MNIST [39] and CIFAR-10 [40]. The results can be found in Section IV.

Finally, we want to mention that prediction uncertainty can be decomposed into two types of uncertainty, namely, epistemic and aleatoric uncertainties. Epistemic uncertainty accounts for uncertainty in model parameters and, thus, is also called model uncertainty as it has been done up until now. This type of uncertainty is covered by the variational posterior distribution. Aleatoric uncertainty, on the other hand, captures noise inherent in the observations (for instance, sensor noise). This type of uncertainty is covered by the distribution used to define the likelihood function. Clearly, an accurate method to measure prediction uncertainty would use sophisticated distributions for the epistemic and for the aleatoric uncertainty. Nevertheless, in this article, we restrict ourselves to propose a good and easy to estimate variational distribution. We use the likelihood function commonly considered in deep learning approaches. More flexible alternatives are proposed in [41]. An extension of our approach including nonstandard likelihood functions will be considered in future work.

II. Variational Inference for Deep Learning

In this section, based on the previous work [20], [27], [31], [38], [42], we briefly discuss how variational inference can be applied in deep learning. Since we are particularly interested in image classification, the focus will be on networks designed for classification tasks. Note that the methodology also holds for regression models after some slight modifications.

Let $\mathbf{W}$ denote the random vector covering all parameters (weights and biases) of a given neural net $\mathbf{f}$. Furthermore, let $p(\mathbf{w})$ denote the density used to define *a priori* knowledge regarding $\mathbf{W}$. According to the Bayesian theorem, the posterior distribution of $\mathbf{W}$ is given by the density

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{w}, \mathbf{X})p(\mathbf{w})}{\int p(\mathbf{y}|\mathbf{w}, \mathbf{X})p(\mathbf{w})\, d\mathbf{w}} \tag{1}$$

where $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_\beta\}$ denotes a set of training examples and $\mathbf{y} = (y_1, \dots, y_\beta)^T$ holds the corresponding class labels. Note that the probability $p(\mathbf{y}|\mathbf{w}, \mathbf{X})$ is given by the product $\prod_{i=1}^{\beta} \mathbf{f}(\mathbf{x}_i; \mathbf{w})_{y_i}$ in accordance with the classical assumptions on stochastic independence and modeling in deep learning for classification. The integral in the abovementioned representation of $p(\mathbf{w}|\mathbf{y}, \mathbf{X})$ is commonly intractable due to its high dimension β. Variational inference aims at approximating the posterior with another parametric distribution, the so-called variational distribution $Q_{\boldsymbol{\phi}}(\mathbf{w})$. To this end, the so-called variational parameters $\boldsymbol{\phi}$ are optimized by minimizing the Kullback–Leibler divergence (KL divergence)

$$D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{w})||p(\mathbf{w}|\mathbf{y}, \mathbf{X})) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{w})}\left(\ln \frac{q_{\boldsymbol{\phi}}(\mathbf{w})}{p(\mathbf{w}|\mathbf{y}, \mathbf{X})}\right) \tag{2}$$

between the variational distribution and the posterior. Although the KL divergence is no formal distance measure (does not satisfy some of the requested axioms), it is a common choice to measure the similarity of probability distributions.

Since the posterior distribution is unknown, the divergence $D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{w})||p(\mathbf{w}|\mathbf{y}, \mathbf{X}))$ cannot be minimized directly. Anyway, the minimization of $D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{w})||p(\mathbf{w}|\mathbf{y}, \mathbf{X}))$ is equivalent to the minimization of the so-called negative log evidence lower bound

$$\begin{aligned} L_{VI} &= -\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{w})}\big[\ln p(\mathbf{y}|\mathbf{w}, \mathbf{X})\big] + D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{w})||p(\mathbf{w})) \\ &= -\sum_{i=1}^{\beta}\big\{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{w})}\big[\ln \mathbf{f}(\mathbf{x}_i; \mathbf{w})_{y_i}\big]\big\} + D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{w})||p(\mathbf{w})). \end{aligned} \tag{3}$$

Thus, the optimization problem reduces to the minimization of the sum of the expected negative log likelihood and the KL divergence between the variational distribution and the prior with respect to $\boldsymbol{\phi}$. Inspired by stochastic gradient descent, it is not unusual to approximate the expected values in L_{VI} via Monte Carlo integration with one sample during network training. Note that the resampling in each training iteration guarantees that a sufficient amount of samples is drawn overall. Commonly, minibatch gradient descent is used for optimization in deep learning. To take account of the resulting reduction of the number of training examples used in each iteration of the optimization, the objective function has to be rescaled. Thus, in the kth iteration, the function to minimize is given by

$$\widehat{L}_{VI} = -\frac{1}{m}\sum_{i=1}^{m}\big\{\ln \mathbf{f}(\tilde{\mathbf{x}}_i; \mathbf{w}_k)_{\tilde{y}_i}]\big\} + \frac{1}{\beta} D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{w})||p(\mathbf{w})) \tag{4}$$

where $\mathbf{w}_k$ denotes a sample from $q_{\boldsymbol{\phi}}(\mathbf{w})$, m denotes the minibatch size, and $\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_m, \tilde{y}_1, \dots, \tilde{y}_m$ denote the minibatch itself.

Summing up, optimizing a Bayesian neural net is quite similar to the optimization of a classical one. However, contrary to frequentist deep learning, where it is common practice to penalize the Euclidean norm of the network parameters for model regularization, Bayesian deep learning penalizes deviations of the variational distribution from the prior. In principle, the same loss function L (cross-entropy loss) is minimized but with the crucial difference that network parameters have to be sampled since they are random.

In Bayesian deep learning, predictions are based on the posterior predictive distribution, i.e., the distribution of a class label y^* for a given example $\mathbf{x}^*$ conditioned on the observed data $\mathbf{y}, \mathbf{X}$. The distribution can be approximated via Monte Carlo integration

$$\begin{aligned} p(y^*|\mathbf{x}^*, \mathbf{y}, \mathbf{X}) &= \int p(y^*|\mathbf{w}, \mathbf{x}^*)p(\mathbf{w}|\mathbf{y}, \mathbf{X})\, d\mathbf{w} \\ &\approx \int p(y^*|\mathbf{w}, \mathbf{x}^*)q_{\boldsymbol{\phi}}(\mathbf{w})\, d\mathbf{w} \\ &\approx \frac{1}{N}\sum_{i=1}^{N}\mathbf{f}(\mathbf{x}^*; \mathbf{w}_i)_{y^*} \end{aligned} \tag{5}$$

where $\mathbf{w}_1, \dots, \mathbf{w}_N$ denote samples from $q_{\boldsymbol{\phi}}(\mathbf{w})$. Therefore, the class of an object $\mathbf{x}^*$ is predicted by computing multiple network outputs with parameters sampled from the variational distribution. Averaging the output vectors results in an estimate of the posterior predictive distribution, such that the *a posteriori* most probable class finally serves as prediction.

The *a posteriori* uncertainty in the network parameters $\mathbf{W}$ directly translates to uncertainty about the random network output $\mathbf{f}(\mathbf{x}^*; \mathbf{W})$ and, thus, the posterior probability of an object $\mathbf{x}^*$ belonging to class y^*. While the posterior predictive distribution incorporates parameter uncertainty by averaging over the respective posterior of the network parameters, credible intervals for the probability that an instance corresponds to a given class can be estimated. Therefore, at first, multiple samples $\mathbf{w}_1, \dots, \mathbf{w}_N$ are drawn from the variational distribution $Q_{\boldsymbol{\phi}}(\mathbf{w})$. Then, the corresponding network outputs $\mathbf{f}(\mathbf{x}^*; \mathbf{w}_1)_{y^*}, \dots, \mathbf{f}(\mathbf{x}^*; \mathbf{w}_N)_{y^*}$ are determined. Finally, the empirical $(1-\alpha)/2$ and $(1+\alpha)/2$ quantiles of these outputs provide an estimate of the α credible interval for the probability $p(y^*|\mathbf{x}^*, \mathbf{y}, \mathbf{X})$.

III. Methodology

In this section, we describe our approach. At first, we give a formal definition of the variational distribution, we use to approximate the posterior, and, additionally, we propose a normal prior. Moreover, we report the derivatives of the approximation $\widehat{L}_{VI}$ of the negative log evidence lower bound (see Section II) with respect to the variational parameters, i.e., the learnable parameters. Finally, we present the pseudocode that is the basis of our implementation of the proposed method.

A. Variational Distribution and Prior

Let $\mathbf{W}_j = (W_{j1}, \dots, W_{jK_j})^T$ denote the random weights of layer j $(j = 1, \dots, d)$. Furthermore, let $\mathbf{B}_j = (B_{j1}, \dots, B_{jk_j})^T$ denote the corresponding random bias terms. The integers K_j and k_j denote the number of weights and the number of biases in layer j, respectively.

As mentioned in Section I, we define the variational distribution as a product of multivariate normal distributions with tridiagonal covariance matrices. Applying variational inference to Bayesian deep learning presupposes that samples from the variational distribution can be drawn during network training as well as at the stage in which new samples are used for prediction. Especially, at the training phase, it is essential that the random sampling can be reduced to the sampling from a (multivariate) standard normal distribution and appropriate affine-linear transformation of the drawn samples based on the learnable parameters. A direct sampling from the nontrivial normal distributions would mask the variational parameters and, thus, make it impossible to optimize them by gradient descent. Provided that a covariance matrix is positive definite (in general, covariance matrices are only positive semidefinite), there exists a unique Cholesky decomposition of the matrix that can be used for this task. Note that for each real-valued symmetric positive-definite square matrix, a unique decomposition (Cholesky decomposition) of the form $\Sigma = \mathbf{L}\mathbf{L}^T$ exists, where L is a lower triangular matrix with real and positive diagonal entries. Thus, we are interested in symmetric tridiagonal matrices that always stay positive definite no matter how the corresponding learnable parameters are adjusted during network training. The first thing required is a criterion for the positive definiteness of for our purposes appropriate tridiagonal matrices. Anđelić and Fonseca [43] gave the following sufficient condition for positive definiteness of tridiagonal matrices: Let

$$\Sigma = \begin{pmatrix} a_1 & c_1 & & & \\ c_1 & a_2 & c_2 & & \\ & c_2 & a_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & c_{n-1} & a_n \end{pmatrix} \in \mathbb{R}^{n\times n}$$

a symmetric tridiagonal matrix with positive diagonal entries. If

$$c_i^2 < \frac{1}{4} a_i a_{i+1} \frac{1}{\cos^2\left(\frac{\pi}{n+1}\right)}, \quad i = 1, \dots, n-1 \tag{6}$$

then Σ is positive definite. Consider now a matrix $\Sigma \in \mathbb{R}^{n\times n}$ defined as follows:

$$\begin{aligned} \Sigma &:= \mathbf{L}\mathbf{L}^T \\ &= \begin{pmatrix} a_1 & & & & \\ c_1 & a_2 & & & \\ & c_2 & a_3 & & \\ & & \ddots & \ddots & \\ & & & c_{n-1} & a_n \end{pmatrix} \begin{pmatrix} a_1 & c_1 & & & \\ & a_2 & c_2 & & \\ & & a_3 & \ddots & \\ & & & \ddots & c_{n-1} \\ & & & & a_n \end{pmatrix} \\ &= \begin{pmatrix} a_1^2 & c_1 a_1 & & & \\ c_1 a_1 & c_1^2 + a_2^2 & c_2 a_2 & & \\ & c_2 a_2 & c_2^2 + a_3^2 & c_3 a_3 & \\ & & \vdots & & \\ & & c_{n-2}a_{n-2} & c_{n-2}^2 + a_{n-1}^2 & c_{n-1}a_{n-1} \\ & & & c_{n-1}a_{n-1} & c_{n-1}^2 + a_n^2 \end{pmatrix}. \end{aligned} \tag{7}$$

If Σ satisfies condition (6) and $\mathbf{L}$ has positive diagonal entries, the matrix $\mathbf{L}$ defines the Cholesky decomposition of Σ, and furthermore, Σ is a valid covariance matrix since every real, symmetric, and positive semidefinite square matrix defines a valid covariance matrix. As in the work of Posch *et al.* [38], we define the variances as multiples of the corresponding squared expectation values, denoted by $m_1^2, \dots, m_n^2$

$$c_{i-1}^2 + a_i^2 := \tau^2 m_i^2, \quad i = 1, \dots, n \tag{8}$$

$$c_0 := 0 \tag{9}$$

where $\tau \in \mathbb{R}^+$ and $m_1, \dots, m_n \in \mathbb{R} \setminus \{0\}$. Defining the variances proportional to the squared expectation values allows for a useful specification of them. This specification requires, besides the expectation values, only one additional variational parameter. Moreover, we want the correlations to be identical, which leads to the following covariances $c_i a_i$:

$$\rho = \frac{c_i a_i}{\sqrt{\tau^2 m_i^2}\sqrt{\tau^2 m_{i+1}^2}} = \frac{c_i a_i}{\tau^2 |m_i||m_{i+1}|} \tag{10}$$

$$\Leftrightarrow c_i a_i = \rho \tau^2 |m_i||m_{i+1}| \tag{11}$$

for $i = 1, \dots, n-1$. By rearranging (8) and (11), one obtains a recursive formula for the elements of the matrix L

$$(11) \Leftrightarrow a_i = \frac{\rho\tau^2 |m_i||m_{i+1}|}{c_i} \tag{12}$$

$$(8) \Leftrightarrow c_{i-1}^2 + \frac{\rho^2 \tau^4 m_i^2 m_{i+1}^2}{c_i^2} = \tau^2 m_i^2 \tag{13}$$

$$\Leftrightarrow c_i^2 = \frac{\rho^2 \tau^4 m_i^2 m_{i+1}^2}{\tau^2 m_i^2 - c_{i-1}^2}. \tag{14}$$

Note that (14), for instance, is satisfied for

$$c_i = \frac{\rho \tau^2 m_i m_{i+1}}{\sqrt{\tau^2 m_i^2 - c_{i-1}^2}}. \tag{15}$$

By defining c_i this way, one does not end up by the Cholesky decomposition that assumes the diagonal elements a_i of L to be positive and, thus by (12), also assumes the product ρc_i to be positive. Replacing the product $m_i m_{i+1}$ by its absolute value in (15) would result in the Cholesky decomposition but is not necessary for our purposes and, therefore, not done. Thus, the elements of $\mathbf{L}$ are recursively defined as

$$c_0 := 0 \tag{16}$$

$$c_i := \frac{\rho\tau^2 m_i m_{i+1}}{\sqrt{\tau^2 m_i^2 - c_{i-1}^2}}, \quad i = 1,\dots,n-1 \tag{17}$$

$$a_i := \frac{\rho\tau^2 |m_i||m_{i+1}|}{c_i}, \quad i = 1,\dots,n-1 \tag{18}$$

$$a_n := \sqrt{\tau^2 m_n^2 - c_{n-1}^2}. \tag{19}$$

Note that the matrix Σ defined by (16)–(19) satisfies condition (6) iff

$$(\rho\tau^2|m_i||m_{i+1}|)^2 < \frac{1}{4}\tau^2 m_i^2 \tau^2 m_{i+1}^2 \frac{1}{\cos^2\left(\frac{\pi}{n+1}\right)} \tag{20}$$

$$\Leftrightarrow \quad \rho^2 < \frac{1}{4}\frac{1}{\cos^2\left(\frac{\pi}{n+1}\right)} \tag{21}$$

$$\Leftrightarrow \quad \rho \in \underbrace{\left(-\frac{1}{2}\frac{1}{\sqrt{\cos^2\left(\frac{\pi}{n+1}\right)}}, \frac{1}{2}\frac{1}{\sqrt{\cos^2\left(\frac{\pi}{n+1}\right)}}\right)}_{\underset{\text{for large } n}{\approx}\left(-\frac{1}{2},\frac{1}{2}\right)}. \tag{22}$$

Thus, provided that condition (22) holds, there exists a unique Cholesky decomposition of Σ which again guarantees that the c_i according to (17) are well defined and, furthermore, also that a_n, according to (19), is well defined.

Using the abovementioned considerations, the variational distribution can finally be defined as follows. In the first step, lower bidiagonal matrices $\mathbf{L}_j$ are specified for $j = 1,\dots,d$

$$\mathbf{L}_j := \begin{pmatrix} a_{j1} & & & & \\ c_{j1} & a_{j2} & & & \\ & c_{j2} & a_{j3} & & \\ & & \ddots & \ddots & \\ & & & c_{j,K_j-1} & a_{jK_j} \end{pmatrix} \tag{23}$$

$$c_{j0} := 0 \tag{24}$$

$$c_{ji} := \frac{\rho_j \tau_j^2 m_{ji} m_{j,i+1}}{\sqrt{\tau_j^2 m_{ji}^2 - c_{j,i-1}^2}}, \quad i = 1,\dots,K_j-1 \tag{25}$$

$$a_{ji} := \frac{\rho_j \tau_j^2 |m_{ji}||m_{j,i+1}|}{c_{ji}}, \quad i = 1,\dots,K_j-1 \tag{26}$$

$$a_{jK_j} := \sqrt{\tau_j^2 m_{jK_j}^2 - c_{j,K_j-1}^2}. \tag{27}$$

The variational distribution of the weights of the jth layer is then defined as a multivariate normal distribution

$$Q_{\boldsymbol{\phi}_j}(\mathbf{w}_j) = \mathcal{N}(\mathbf{w}_j; \mathbf{m}_j, \Sigma_j) \tag{28}$$

with expected value $\mathbf{m}_j$ and a tridiagonal covariance matrix $\Sigma_j = \mathbf{L}_j\mathbf{L}_j^T$. According to the abovementioned considerations, the variances of the normal distribution are given by $\tau_j^2 m_{ji}^2$ $(i = 1,\dots,K_j)$, and the covariances are given by $\rho_j\tau_j^2|m_{ji}||m_{j,i+1}|$ $(i = 1,\dots,K_j-1)$. This again implies that the correlations are all the same and given by the parameter ρ_j. Since the parameter τ_j regulates the variances of the distribution, it should not take negative values during optimization. To guarantee this, it is reparameterized with help of the softplus function

$$\tau_j = \ln(1+\exp(\delta_j)) > 0. \tag{29}$$

Moreover, the parameter ρ_j should lie in the interval $(-1/2, 1/2)$ to ensure that the matrix Σ_j is positive definite. In deep learning, dimensions are commonly high, such that the approximation in (22) can be considered as valid. The following reparameterization ensures that the desired property holds:

$$\rho_j = \frac{1}{1+\exp(-\gamma_j)} - \frac{1}{2} \in \left(-\frac{1}{2}, \frac{1}{2}\right). \tag{30}$$

In addition, the diagonal entries of Σ_j have to be nonzero to ensure positive definiteness, which again implies that each component of $\mathbf{m}_j$ has to be nonzero. We decide to set values that are not significantly different from zero to small random numbers in our implementation instead of introducing another reparameterization. Finally, $\mathbf{m}_j \in \mathbb{R}^{K_j} \setminus \{\mathbf{0}\}$, $\delta_j \in \mathbb{R}$, and $\gamma_j \in \mathbb{R}$ can be summarized as the variational parameters $\boldsymbol{\phi}_j$ corresponding to the weights of the jth network layer.

One can easily sample from a random vector $\mathbf{W}_j$ belonging to this distribution using samples from a standard normal distribution $\mathcal{N}(0,1)$ since it can be written as

$$\mathbf{W}_j = \mathbf{m}_j + \mathbf{L}_j\mathbf{X}_j \text{ with } \mathbf{X}_j \sim \mathcal{N}(\mathbf{0}_{K_j}, \mathbf{I}_{K_j}). \tag{31}$$

Note that (31) can also be written as

$$W_{j1} = m_{j1} + a_{j1}X_{j1} \tag{32}$$

$$\vdots$$

$$W_{ji} = m_{ji} + c_{j,i-1}X_{j,i-1} + a_{ji}X_{ji} \tag{33}$$

$$\vdots$$

$$W_{jK_j} = m_{jK_j} + c_{j,K_j-1}X_{j,K_j-1} + a_{jK_j}X_{jK_j}. \tag{34}$$

The layerwise variational distributions of the bias terms denoted by $q_{\boldsymbol{\phi}_{bj}}(\mathbf{b}_j)$ are defined completely analogous to those of the weights. Assuming independence of the layers as well as independence between weights and biases, the overall variational distribution is given by

$$q_{\boldsymbol{\phi}}(\mathbf{w}) = \prod_{j=1}^{d} q_{\boldsymbol{\phi}_j}(\mathbf{w}_j) q_{\boldsymbol{\phi}_{bj}}(\mathbf{b}_j) \tag{35}$$

where $\boldsymbol{\phi}_j = \{\mathbf{m}_j, \delta_j, \gamma_j\}$, $\boldsymbol{\phi}_{bj} = \{\mathbf{m}_{bj}, \delta_{bj}, \gamma_{bj}\}$, $q_{\boldsymbol{\phi}_j}(\mathbf{w}_j)$ denotes the density of $\mathcal{N}(\mathbf{m}_j, \Sigma_j)$, $q_{\boldsymbol{\phi}_{bj}}(\mathbf{b}_j)$ denotes the density of $\mathcal{N}(\mathbf{m}_{bj}, \Sigma_{bj})$, and $\mathbf{w}$ is a vector, including all weights and all biases.

We define the *a priori* distribution completely analogous to Posch *et al.* [38]. In particular, its density is given by

$$p(\boldsymbol{w}) = \prod_{j=1}^{d} p(\boldsymbol{w}_j)p(\boldsymbol{b}_j) \tag{36}$$

where $p(\boldsymbol{w}_j)$ denotes the density of $N(\boldsymbol{\mu}_j, \zeta_j^2\mathbf{I}_{K_j})$ and $p(\boldsymbol{b}_j)$ denotes the density of $N(\boldsymbol{\mu}_{bj}, \zeta_{bj}^2\mathbf{I}_{k_j})$.

B. Kullback–Leibler Divergence

The fact that the variational distribution as well as the prior factorize simplify the computation of the KL divergence. Indeed, the overall divergence is given by the sum of the layerwise divergences (for further details, refer to [38])

$$D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{w})||p(\mathbf{w})) = \sum_{j=1}^{d}\Big[D_{KL}(q_{\boldsymbol{\phi}_j}(\mathbf{w}_j)||p(\mathbf{w}_j))\Big] + \sum_{j=1}^{d}\Big[D_{KL}(q_{\boldsymbol{\phi}_{bj}}(\mathbf{b}_j)||p(\mathbf{b}_j))\Big]. \quad (37)$$

Thus, computing the overall divergence can be reduced to compute $D_{KL}(q_{\boldsymbol{\phi}_j}(\mathbf{w}_j)||p(\mathbf{w}_j))$ for fixed $j \in \{1,\dots,K_j\}$ since the remaining divergences compute completely analogously (only the indices differ). According to Hershey and Olsen [44], the KL divergence between two p-dimensional normal distributions, given by $H(\mathbf{x}) = \mathcal{N}(\mathbf{x};\boldsymbol{\mu}_h,\Sigma_h)$ and $G(\mathbf{x}) = \mathcal{N}(\mathbf{x};\boldsymbol{\mu}_g,\Sigma_g)$, computes as

$$D_{KL}(H||G) = \frac{1}{2}\bigg[\ln\frac{|\Sigma_g|}{|\Sigma_h|} + \operatorname{tr}(\Sigma_g^{-1}\Sigma_h) - p + (\boldsymbol{\mu}_h - \boldsymbol{\mu}_g)^T\Sigma_g^{-1}(\boldsymbol{\mu}_h - \boldsymbol{\mu}_g)\bigg]. \quad (38)$$

Thus, the determinant of the covariance matrix Σ_j is required for the computation of the KL divergence $D_{KL}(q_{\boldsymbol{\phi}_j}(\mathbf{w}_j)||p(\mathbf{w}_j))$. Using basic properties of determinants $|\Sigma_j|$ computes as follows for fixed j:

$$|\Sigma_j| = \big|\mathbf{L}_j\mathbf{L}_j^T\big| = |\mathbf{L}_j|\big|\mathbf{L}_j^T\big| = |\mathbf{L}|^2 = \prod_{i=1}^{K_j} a_{ji}^2. \quad (39)$$

Using (38) and (39) $D_{KL}(q_{\boldsymbol{\phi}_j}(\mathbf{w}_j)||p(\mathbf{w}_j))$ then reads

$$\begin{aligned} &D_{KL}(q_{\boldsymbol{\phi}_j}(\mathbf{w}_j)||p(\mathbf{w}_j))\\ &= \frac{1}{2}\Bigg[\ln\frac{\big|\zeta_j^2\mathbf{I}_{K_j}\big|}{|\Sigma_j|} + \operatorname{tr}\big((\zeta_j^2\mathbf{I}_{K_j})^{-1}\Sigma_j\big) - K_j + (\mathbf{m}_j - \boldsymbol{\mu}_j)^T\big(\zeta_j^2\mathbf{I}_{K_j}\big)^{-1}(\mathbf{m}_j - \boldsymbol{\mu}_j)\Bigg]\\ &= \frac{1}{2}\Bigg[-\ln\Bigg(\prod_{i=1}^{K_j}a_{ji}^2\Bigg) + \frac{\tau_j^2}{\zeta_j^2}||\mathbf{m}_j||_2^2 + \frac{1}{\zeta_j^2}||\mathbf{m}_j - \boldsymbol{\mu}_j||_2^2 + c\Bigg]\\ &= \frac{1}{2}\Bigg[-\sum_{i=1}^{K_j}\big(\ln a_{ji}^2\big) + \frac{\tau_j^2}{\zeta_j^2}||\mathbf{m}_j||_2^2 + \frac{1}{\zeta_j^2}||\mathbf{m}_j - \boldsymbol{\mu}_j||_2^2 + c\Bigg] \end{aligned} \quad (40)$$

where c always denotes an additive constant.

C. Derivatives

Commonly, neural networks are optimized via minibatch gradient descent. Thus, in order to train a neural net $\mathbf{f}(\cdot,\mathbf{w})$, according to our approach, the partial derivatives of the approximation $\widehat{L}_{VI}$ of the negative log evidence lower bound described in Section II with respect to the variational parameters $\boldsymbol{\phi}_j = \{\mathbf{m}_j,\delta_j,\gamma_j\}, \boldsymbol{\phi}_{bj} = \{\mathbf{m}_{bj},\delta_{bj},\gamma_{bj}\}$ are required. In particular, the partial derivatives of the loss function L typically used in deep learning and the partial derivatives of the KL divergence between prior and variational distribution have to be computed. Note that the loss function equals the negative log likelihood of the data and is given by the cross-entropy loss in the case of classification and by the Euclidean loss in the case of regression. Thus, L depends on the network $\mathbf{f}$ itself, with parameters sampled from the variational distribution $q_{\boldsymbol{\phi}}(\mathbf{w})$. With the help of the multivariate chain rule, the required partial derivatives of L can be computed based on the classical derivatives used in non-Bayesian deep learning

$$\frac{\partial L}{\partial \mathbf{m}_j} = \left(\frac{\partial \mathbf{w}_j}{\partial \mathbf{m}_j}\right)^T\frac{\partial L}{\partial \mathbf{w}_j} \Rightarrow \frac{\partial L}{\partial m_{ji}} = \sum_l \frac{\partial L}{\partial w_{jl}}\frac{\partial w_{jl}}{\partial m_{ji}} \quad (41)$$

$$\frac{\partial L}{\partial \delta_j} = \left(\frac{\partial \mathbf{w}_j}{\partial \delta_j}\right)^T\frac{\partial L}{\partial \mathbf{w}_j} \Rightarrow \frac{\partial L}{\partial \delta_j} = \sum_l \frac{\partial L}{\partial w_{jl}}\frac{\partial w_{jl}}{\partial \delta_j} \quad (42)$$

$$\frac{\partial L}{\partial \gamma_j} = \left(\frac{\partial \mathbf{w}_j}{\partial \gamma_j}\right)^T\frac{\partial L}{\partial \mathbf{w}_j} \Rightarrow \frac{\partial L}{\partial \gamma_j} = \sum_l \frac{\partial L}{\partial w_{jl}}\frac{\partial w_{jl}}{\partial \gamma_j}. \quad (43)$$

Equations (41)–(43) only deal with the derivatives of L with respect to the variational parameters belonging to the network weights. Completely analogous equations hold for the bias terms. In the sequel, we focus on the derivatives of the weights since the derivatives for the biases are obviously of the same form. Note that for a given sample $\mathbf{w}$ from the variational distribution, the layerwise derivatives $(\partial L/\partial \mathbf{w}_j), (\partial L/\partial \mathbf{b}_j)$ $(j = 1,\dots,d)$ are computed as in non-Bayesian deep learning. Thus, the problem of finding closed form expressions for the required derivatives of L reduces to the problem of finding these expressions for the $\mathbf{w}_j$'s and the $\mathbf{b}_j$'s. Taking account of (32)–(34), the needed derivatives of the weights can be expressed in terms of the corresponding derivatives of the c_{ji} and the a_{ji}

$$\frac{\partial w_{ji}}{\partial m_{jk}} = \begin{cases} \dfrac{\partial c_{j,i-1}}{\partial m_{jk}}x_{j,i-1} + \dfrac{\partial a_{ji}}{\partial m_{jk}}x_{ji}, & k \neq i\\ 1 + \dfrac{\partial c_{j,i-1}}{\partial m_{ji}}x_{j,i-1} + \dfrac{\partial a_{ji}}{\partial m_{ji}}x_{ji}, & k = i \end{cases} \quad (44)$$

$$\frac{\partial w_{ji}}{\partial \delta_j} = \frac{\partial c_{j,i-1}}{\partial \delta_j}x_{j,i-1} + \frac{\partial a_{ji}}{\partial \delta_j}x_{ji} \quad (45)$$

$$\frac{\partial w_{ji}}{\partial \gamma_j} = \frac{\partial c_{j,i-1}}{\partial \gamma_j}x_{j,i-1} + \frac{\partial a_{ji}}{\partial \gamma_j}x_{ji} \quad (46)$$

where the index j lies in the set $\{1,\dots,d\}$, while for a given j, the indices i and k lie in the set $\{1,\dots,K_j\}$. The derivatives of the c_{ji} $(i = 1,\dots,K_j - 1)$ with respect to the variational parameters are given by

$$\frac{\partial c_{ji}}{\partial m_{jk}} = \begin{cases} \big(\rho_j\tau_j^2 m_{ji}m_{j,i+1}\big)u^{-\frac{3}{2}}c_{j,i-1}\dfrac{\partial c_{j,i-1}}{\partial m_{jk}}, & k < i\\ \dfrac{\upsilon\Big[\sqrt{u} - m_{ji}u^{-\frac{1}{2}}\Big(\tau_j^2 m_{ji} - c_{j,i-1}\frac{\partial c_{j,i-1}}{m_{ji}}\Big)\Big]}{u}, & k = i\\ \dfrac{\rho_j\tau_j^2 m_{ji}}{\sqrt{u}}, & k = i+1\\ 0, & k > i+1 \end{cases} \quad (47)$$

$$\frac{\partial c_{ji}}{\partial \delta_j} = w_\delta \frac{\upsilon m_{ji}\Big[2\sqrt{u} - \tau_j u^{-\frac{1}{2}}\Big(\tau_j m_{ji}^2 - c_{j,i-1}\frac{\partial c_{j,i-1}}{\partial \tau_j}\Big)\Big]}{\tau_j u} \tag{48}$$

$$\frac{\partial c_{ji}}{\partial \gamma_j} = w_\gamma \frac{\tau_j^2 m_{ji} m_{j,i+1}\Big[\sqrt{u} - \rho_j u^{-\frac{1}{2}}\Big(-c_{j,i-1}\frac{\partial c_{j,i-1}}{\partial \rho_j}\Big)\Big]}{u} \tag{49}$$

where $u := \tau_j^2 m_{ji}^2 - c_{j,i-1}^2$, $\upsilon := \rho_j \tau_j^2 m_{j,i+1}$, $w_\delta := \exp(\delta_j)/[1+\exp(\delta_j)]$, and $w_\gamma := \exp(-\gamma_j)/[1+\exp(-\gamma_j)]^2$, and obviously, each derivative of c_{j0} equals zero. Moreover, the derivatives of the a_{ji} $(i = 1, \ldots, K_j - 1)$ with respect to the variational parameters are given by

$$\frac{\partial a_{ji}}{\partial m_{jk}} = \begin{cases} -\big(\rho_j \tau_j^2 |m_{ji}||m_{j,i+1}|\big) c_{ji}^{-2} \dfrac{\partial c_{ji}}{\partial m_{jk}}, & k < i \\ \dfrac{\rho_j \tau_j^2 |m_{j,i+1}|\Big[\operatorname{sign}(m_{ji}) c_{ji} - |m_{ji}| \frac{\partial c_{ji}}{\partial m_{ji}}\Big]}{c_{ji}^2}, & k = i \\ \dfrac{\rho_j \tau_j^2 |m_{j,i}|\Big[\operatorname{sign}(m_{j,i+1}) c_{ji} - |m_{j,i+1}| \frac{\partial c_{ji}}{\partial m_{j,i+1}}\Big]}{c_{ji}^2}, & k = i+1 \\ 0, & k > i+1 \end{cases} \tag{50}$$

$$\frac{\partial a_{ji}}{\partial \delta_j} = w_\delta \frac{\rho_j \tau_j |m_{ji}||m_{j,i+1}|\Big[2c_{ji} - \tau_j \frac{\partial c_{ji}}{\partial \tau_j}\Big]}{c_{ji}^2} \tag{51}$$

$$\frac{\partial a_{ji}}{\partial \gamma_j} = w_\gamma \frac{\tau_j^2 |m_{ji}||m_{j,i+1}|\Big[c_{ji} - \rho_j \frac{\partial c_{ji}}{\partial \rho_j}\Big]}{c_{ji}^2}. \tag{52}$$

In addition, the derivatives of a_{jK_j} with respect to the variational parameters are given by

$$\frac{\partial a_{jK_j}}{\partial m_{jk}} = \begin{cases} -y^{-\frac{1}{2}} c_{j,K_j-1} \dfrac{\partial c_{j,K_j-1}}{\partial m_{jk}}, & k < K_j \\ y^{-\frac{1}{2}}\Big(\tau_j^2 m_{jK_j} - c_{j,K_j-1}\dfrac{\partial c_{j,K_j-1}}{\partial m_{jK_j}}\Big), & k = K_j \end{cases} \tag{53}$$

$$\frac{\partial a_{jK_j}}{\partial \delta_j} = w_\delta y^{-\frac{1}{2}}\Big(\tau_j m_{jK_j}^2 - c_{j,K_j-1}\frac{\partial c_{j,K_j-1}}{\partial \tau_j}\Big) \tag{54}$$

$$\frac{\partial a_{jK_j}}{\partial \gamma_j} = w_\gamma y^{-\frac{1}{2}}\Big(-c_{j,K_j-1}\frac{\partial c_{j,K_j-1}}{\partial \rho_j}\Big) \tag{55}$$

where $y := \tau_j^2 m_{jK_j}^2 - c_{j,K_j-1}^2$.

Finally, the partial derivatives of the KL divergence $D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{w})||p(\mathbf{w}))$ (abbreviated with D_{KL}) with respect to the variational parameters are given by

$$\frac{\partial}{\partial m_{jk}} D_{KL} = -\sum_{i=1}^{K_j}\Big(\frac{1}{a_{ji}}\frac{\partial a_{ji}}{\partial m_{jk}}\Big) + \frac{\tau_j^2}{\zeta_j^2} m_{jk} + \frac{1}{\zeta_j^2}(m_{jk} - \mu_{jk}) \tag{56}$$

$$\frac{\partial}{\partial \delta_j} D_{KL} = -\sum_{i=1}^{K_j}\Big(\frac{1}{a_{ji}}\frac{\partial a_{ji}}{\partial \delta_j}\Big) + \frac{\exp(\delta_j)}{1+\exp(\delta_j)}\frac{\tau_j}{\zeta_j^2}||\mathbf{m}_j||_2^2 \tag{57}$$

$$\frac{\partial}{\partial \gamma_j} D_{KL} = -\sum_{i=1}^{K_j}\Big(\frac{1}{a_{ji}}\frac{\partial a_{ji}}{\partial \gamma_j}\Big). \tag{58}$$

For reasons of brevity, the calculation process corresponding to the derivatives presented is not reported.

D. Implementation and Pseudocode

We have implemented the proposed approach by modifying and extending the popular open-source deep learning framework Caffe (see [45]). In particular, our implementation includes a Bayesian version of the classical inner product layer as well as the convolutional layer. This allows for the training of (deep) multilayer perceptrons (MLPs) and, moreover, CNNs according to our approach. Up until now, we have not parallelized our code such that it can run on GPU. This is left for future research and will enable the training of state-of-the-art CNNs for image classification in a reasonable amount of time. The pseudocode that was the starting point of our implementation is presented in the following. The code shows how a classical, i.e., frequentist, inner product, or convolutional layer can be extended in order to fit with the methodology presented.

Besides parameters, for which obviously initial values are required, the presented pseudocode also asks for the two additional parameters ν and κ. Empirically, we discovered that in practical applications, it can be helpful to use another penalization strength of the KL divergence than the fixed one proposed in Section II. In particular, the derivatives of the KL divergence do not suffer from the vanishing gradients problem in contrast to the derivatives of the loss function. For this reason, reducing the penalization strength of the divergence by a fixed factor, which is equivalent to reducing the learning rate of the divergence, might be a good decision. Moreover, again, empirically, we discovered that the derivatives with respect to the γ_j are quite small, which slows down the learning process. To overcome this problem, the learning rate multiplier κ can be used.

In the forward pass, at first, variational parameters that hold "critical" values are assigned similar but less "critical" ones. This way of proceeding guarantees that no numerical issues occur during the training process. Thus, very small expectation values, variances (smaller than 0.01 times the absolute value of the corresponding expectation value), and correlations (smaller than 0.01 in absolute values) are replaced, as well as correlations that are nearly given by 0.5 or −0.5. Recall that the correlations have to stay in the interval $(-1/2, 1/2)$ in order for the covariance matrices to be positive definite. After checking for numerical stability, the weights and the biases of the network are sampled from the variational distribution. The sampled parameters are then used in place of the deterministic ones in non-Bayesian deep learning in order to perform the classical forward pass.

In the backward pass, at first, the derivatives of the loss function with respect to the sampled weights and biases are computed. This is done completely analogously as in classical deep learning, but the sampled parameters are used in place of the deterministic ones. In the next steps, the derivatives of the sampled network parameters with respect to the variational parameters are computed, and the derivatives of the KL divergence with respect to the variational parameters are calculated. Finally, an appropriate merging of all the computed derivatives results in the derivatives of the approximation $\widehat{L}_{VI}$ of the negative log evidence lower bound. These derivatives are used to update the variational parameters according to some learning schedule.

Due to the tridiagonal covariance structure, the proposed variational distribution (see Section III-A) only correlates neighbored parameters. Thus, the order of the components within the layerwise weight vectors is a critical aspect of the definition of this distribution. In our implementation, the order is given as follows. In the case of an inner product layer, the first vector element is given by the first weight of the first neuron, followed by the second weight of the first neuron and so on until the last weight of the last neuron defines the last vector element. For the convolutional layers, the weight vectors are filled by rowwise going through the channels of the kernels (one channel to the next, and one kernel to the next).

IV. Performance Evaluation

This section investigates how well the proposed approach (Gauss cor.) performs on real-world data sets compared with other methods. In particular, the performance comparison includes the frequentist approach, the proposed approach but without correlations (Gauss ind.) [38], and, finally, the popular approach that applies dropout before every weight layer in terms of a Bernoulli variational distribution (Bernoulli) [15]. On the one hand, the prediction accuracy and, on the other hand, the usefulness and quality of the uncertainty information provided are of particular interest. The popular benchmark data sets MNIST [39] and CIFAR-10 [40] form basis of the evaluations.

A. MNIST

In this section, the performance of the proposed method is evaluated based on the MNIST data set. This data set contains 70 000 grayscale images of handwritten digits (0–9). The complete data set is partitioned in a training data set that counts 60 000 images and a test set of 10 000 examples. Each image is of size 28×28.

The architecture chosen for the performance evaluation is the popular LeNet proposed by Lecun *et al.* [46]. This CNN mainly consists of two convolutional layers and, furthermore, two fully connected layers. The special version of LeNet used in this article has the following additional specifications. The ReLU activation function is assigned to the first fully connected layer, while the other layers simply use the identity function as the activation function. Moreover, the first convolutional layer includes 20, and the second one includes 50, 5×5 kernels. Max-pooling with kernel size 2×2 and stride of 2 is applied after both convolutional layers. The number of neurons of the second fully connected layer is determined by the number of possible classes and, thus, given by 10. In contrast to the second fully connected layer, the number of neurons in the first fully connected layer can be freely specified. In our experiments, we set this number once to 100 and once to 250.

Pseudocode Bayes Deep Learning Layer

```
Require:
1:    - Initial variational parameters $\mathbf{m}_j$, $\delta_j$, $\gamma_j$, $\mathbf{b}_j$, $\delta_{bj}$
        and $\gamma_{bj}$
      - Factor $\nu$ for penalization strength of $D_{KL}$
      - Learning rate multiplier $\kappa$ for $\gamma_j$
      - Parameters $\boldsymbol{\mu}_j, \zeta_j, \boldsymbol{\mu}_{bj}$ and $\zeta_{bj}$ for the prior
      - Number of training iterations $N_{iter}$
2: for $r$ in $1 : N_{iter}$ do
3:
4:   Forward pass:
5:
6:   Guarantee numerical stability:
7:   if $\gamma_j \in (-0.04000533, 0.04000533)$ then
8:     Set $\gamma_j$ to 0.04000533 with probability $\frac{1}{2}$ and to
9:       $-0.04000533$ otherwise
10:  end if
11:  if $\gamma_j > 10$ then
12:    Set $\gamma_j = 10$
13:  end if
14:  if $\gamma_j < -10$ then
15:    Set $\gamma_j = -10$
16:  end if
17:  if $\delta_j < -4.600166$ then
18:    Set $\delta_j = -4.600166$
19:  end if
20:  for $k$ in $1 : K_j$ do
21:    if $m_{jk} \in (-0.000001, 0.000001)$ then
22:      Set $m_{jk}$ to 0.000001 with probability $\frac{1}{2}$ and to
23:        $-0.000001$ otherwise
24:    end if
25:  end for
26:
27:  Sample from the variational distribution:
28:  Draw $K_j$ independent samples from $\mathcal{N}(0, 1)$ and
     thus a sample $\mathbf{x}_j$ from $\mathcal{N}(\mathbf{0}_{K_j}, \mathbf{I}_{K_j})$
29:  Compute $\mathbf{L}_j$ according to Equations (23-27, 29-30)
30:  Set $\mathbf{w}_j = \mathbf{m}_j + \mathbf{L}_j\mathbf{x}_j$
31:
32:  Repeat lines $(6 - 30)$, but now for the biases
33:  Use the the sampled weights and biases in place of
     the classical weights and biases in non-Bayesian
     deep learning and proceed as in the classical case
34:
35:  Backward pass
36:
37:  Treat the sampled weights as the classical ones in
     non-Bayesian deep learning in order to compute the
     derivatives of the loss function $L$ with respect to
     them
```

Pseudocode *(Continued.)* Bayes Deep Learning Layer

```
38: *Compute the derivatives of $\widehat{L}_{VI}$ with respect to $\mathbf{m}_j$:*
39: Define: $a_{Deriv} = 0, c_{NewDeriv} = 0,\ c_{OldDeriv} = 0$
40: Declare a variable $wDeriv$
41: Declare a $K_j$ dimensional array diff$_{m_j}$, which will
    hold the derivative $\frac{\partial \widehat{L}_{VI}}{\partial m_{jk}}$ as $k$-th entry
42: Initialize diff$_{m_j}$ with zeros
43: **for** $k$ in $1 : K_j$ **do**
44:   **for** $l$ in $1 : K_j - 1$ **do**
45:     Assign the derivative $\frac{\partial c_{jl}}{\partial m_{jk}}$ according to
        Equation (47) to $c_{NewDeriv}$ (note: $c_{OldDeriv}$
        holds $\frac{\partial c_{j,l-1}}{\partial m_{jk}}$)
46:     Assign the derivative $\frac{\partial a_{jl}}{\partial m_{jk}}$ given by
        Equation (50) to $a_{Deriv}$
47:     Assign the derivative $\frac{\partial w_{jl}}{\partial m_{jk}}$ according to
        Equation (44) to $w_{Deriv}$
48:     In compliance with Equation (41) add $\frac{\partial L}{\partial w_{jl}} w_{Deriv}$
        to diff$_{m_j}[k]$
49:     In accordance with Equation (56) subtract $\nu \frac{a_{Deriv}}{a_{jl}}$
        from diff$_{m_j}[k]$
50:     Set $c_{OldDeriv} = c_{NewDeriv}$
51:   **end for**
52:   Set $l = K_j$
53:   Assign the derivative $\frac{\partial a_{jl}}{\partial m_{jk}}$ according to
      Equation (53) to $a_{Deriv}$
54:   Assign the derivative $\frac{\partial w_{jl}}{\partial m_{jk}}$ given by
      Equation (44) to $w_{Deriv}$
55:   In compliance with (41) add $\frac{\partial L}{\partial w_{jl}} w_{Deriv}$ to
      diff$_{m_j}[k]$
56:   In accordance with Equation (56) subtract $\nu \frac{a_{Deriv}}{a_{jl}}$
      from diff$_{m_j}[k]$ and add the term
      $\nu \left[ \frac{\tau_j^2}{\zeta_j^2} m_{jk} + \frac{1}{\zeta_j^2}(m_{jk} - \mu_{jk}) \right]$
57: **end for**
58:
59: *Compute the derivatives of $\widehat{L}_{VI}$ with respect to $\delta_j$:*
60: Calculate the derivatives $\frac{\partial c_{ji}}{\partial \delta_j}$ $(i = 1, \ldots, K_j - 1)$
    according to Equation (48)
61: Determine the derivatives $\frac{\partial a_{ji}}{\partial \delta_j}$ $(i = 1, \ldots, K_j)$
    given by the Equations (51) and (54)
62: Evaluate the derivatives $\frac{\partial w_{ji}}{\partial \delta_j}$ $(i = 1, \ldots, K_j)$
    in accordance with Equation (45)
63: Find out the derivative $\frac{\partial L}{\partial \delta_j}$ according to Equation
    (42)
64: In order to finally obtain $\frac{\partial \widehat{L}_{VI}}{\partial \delta_j}$ add $\nu \frac{\partial}{\partial \delta_j} D_{KL}$
    in compliance with Equation (57) to $\frac{\partial L}{\partial \delta_j}$
```

Pseudocode *(Continued.)* Bayes Deep Learning Layer

```
65: *Compute the derivatives of $\widehat{L}_{VI}$ with respect to $\gamma_j$:*
66: Calculate the derivatives $\frac{\partial c_{ji}}{\partial \gamma_j}$ $(i = 1, \ldots, K_j - 1)$
    according to Equation (49)
67: Determine the derivatives $\frac{\partial a_{ji}}{\partial \gamma_j}$ $(i = 1, \ldots, K_j)$
    given by the Equations (52) and (55)
68: Evaluate the derivatives $\frac{\partial w_{ji}}{\partial \gamma_j}$ $(i = 1, \ldots, K_j)$
    in accordance with Equation (46)
69: Find out the derivative $\frac{\partial L}{\partial \gamma_j}$ according to Equation
    (43)
70: In order to finally obtain $\frac{\partial \widehat{L}_{VI}}{\partial \gamma_j}$ add $\nu \frac{\partial}{\partial \gamma_j} D_{KL}$
    in compliance with Equation (58) to $\frac{\partial L}{\partial \gamma_j}$
71: Multiply $\frac{\partial \widehat{L}_{VI}}{\partial \gamma_j}$ with $\kappa$
72:
73: Determine the gradient with respect to bottom data
    as it is done in frequentist deep learning, but use
    the sampled weights $\mathbf{w}_j$ in place of the deterministic
    ones
74: Repeat lines $(38 - 73)$, but now for the biases
75: Use the computed derivatives of $\widehat{L}_{VI}$ with respect
    to the variational parameters in order to update
    them according to some chosen learning schedule
76:
77: **end for**
```

Minibatch gradient descent with a batch size of 64 is chosen for optimization. Furthermore, a decreasing learning rate policy is selected. In particular, the learning rate in the kth iteration is specified as $0.01 \cdot (1 + 0.0001 \cdot i)^{-0.75}$. Moreover, momentum is applied with a value of 0.9. The total amount of iterations is set to 100 000. In terms of regularization in the frequentist approach a penalization of the Euclidean norm with a penalization strength of 0.0005 takes place. Furthermore, dropout is applied after the first fully connected layer with a dropping rate given by 0.5. These regularization techniques are not applied in the proposed Bayesian approach that is naturally regularized due to the sampling from the variational distribution and the penalization of the KL divergence during network training. However, in Bayesian deep learning, the parameters of the prior have to be specified. We set the expectation values of all network parameters to 0 and the variances to 1. There is no *a priori* knowledge available such that the prior can merely be used to guarantee that the network parameters do not diverge. Moreover, for each network layer, we set the parameters ν and κ described in Section III-D to $1/(60\,000 \cdot 100)$ and 50, respectively. Thus, the penalization strength of the KL divergence is reduced by a factor of 100 due to the reasons described in Section III-D. For the Bayesian method without correlations (Gauss ind.), the prior and also the specification of ν are chosen identically to the specifications for the proposed approach (Gauss cor.). Within the Bernoulli variational distribution, a hand-tuned dropping rate of 0.2 is used.

For the model with 100 neurons in the first fully connected layer, the frequentist training process and, furthermore, the training process corresponding to the proposed Bayesian approach (Gauss cor.) are visualized in Figs. 1 and 2. One can see that the loss decreases in a quite similar way in both processes. This is an interesting result since in the work of Posch *et al.* (Gauss ind.) [38], the Bayesian loss fluctuates heavily compared with the classical one. Thus, the learning of correlations between network parameters (which is our

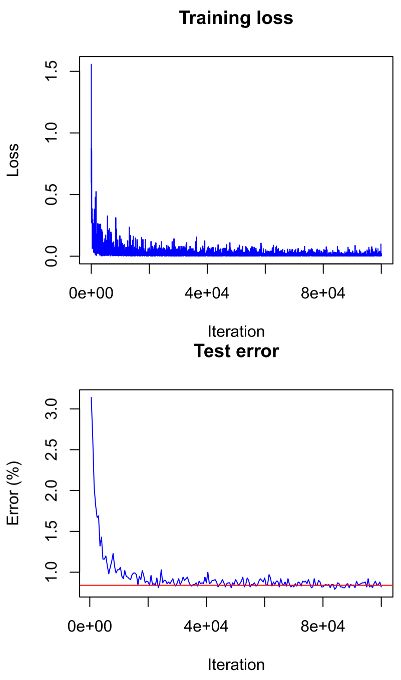


Fig. 1. Training visualization of frequentist LeNet with 100 neurons in the first fully connected layer. The horizontal line marks the achieved test error.

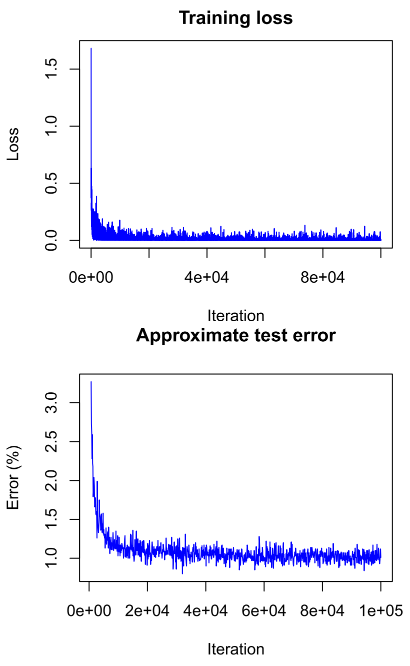


Fig. 2. Training visualization of the proposed approach (Gauss cor.) with 100 neurons in the first fully connected layer. For the computation of the approximate test error, only one sample is drawn from the variational distribution per image.

TABLE I
TEST ERRORS OF THE TRAINED MODELS

Model	Neurons	Test error
Frequentist	100	0.84%
Gauss cor.	100	0.70%
Gauss ind.	100	1.05%
Bernoulli	100	0.78%
Frequentist	250	0.70%
Gauss cor.	250	0.61%
Gauss ind.	250	1.00%
Bernoulli	250	0.78%

TABLE II
VARIATIONAL PARAMETERS ρ_j AND τ_j $(j = 1, \ldots, 4)$

Layer	Neurons	τ_j	τ_{bj}	ρ_j	ρ_{bj}
Convolutional 1	100	0.04	0.05	-0.44	-0.04
Convolutional 2	100	0.52	0.05	-0.25	-0.01
Fully connected 1	100	0.86	0.05	-0.15	0.01
Fully connected 2	100	0.05	0.04	-0.21	0.01
Convolutional 1	250	0.03	0.05	-0.44	0.03
Convolutional 2	250	0.35	0.05	-0.21	-0.01
Fully connected 1	250	2.02	0.06	-0.15	-0.01
Fully connected 2	250	0.06	0.05	-0.18	0.01

extension of their work) enables smoother network training. Note that the test error plotted in Fig. 2 is just a rough approximation of the true one, which is based on one sample from the variational distribution per test image. Usually, the corresponding true errors are significantly lower. However, the rough approximation suffices to monitor network training. A more accurate and, thus, also computationally more expensive approximation based on multiple samples can be made at the end of the training process.

For all considered models, the finally achieved test errors are given in Table I. Note that the predictions of the Bayesian models are based on 200 samples from the corresponding variational distributions per test image. One can see that our approach (Gauss cor.) performs slightly better than the other ones. Especially, the significant improvement compared with the Gaussian approach without correlations (Gauss ind.) shows that allowing for dependencies between parameters pays off.

In Table II, the correlations ρ_j and the variance determining parameters τ_j of the two models corresponding to our approach can be found $(j = 1, \ldots, 4)$. One can see that there is a low *a posteriori* uncertainty about the bias terms and, furthermore, that they are nearly uncorrelated. This is a plausible result since the number of bias terms is small compared with the number of weights. The *a posteriori* uncertainty of the weights differs significantly from layer to layer. While the first convolutional layer and the second fully connected layer go along with a low *a posteriori* uncertainty about the network weights, the other layers show a high uncertainty. For the network with 250 neurons in the first fully connected layer, the standard deviation of the posterior distribution of a weight is given by two times the corresponding expectation value (in absolute values). Note that layers with low uncertainty are exactly the ones that directly act on the network input and the network output. The correlations of the weights are all negative.

As already mentioned in Section II, the uncertainties about the network parameters of the Bayesian nets can directly be translated to uncertainty about the predictions. While the posterior predictive distribution incorporates parameter uncertainty by averaging over the respective posterior, credible

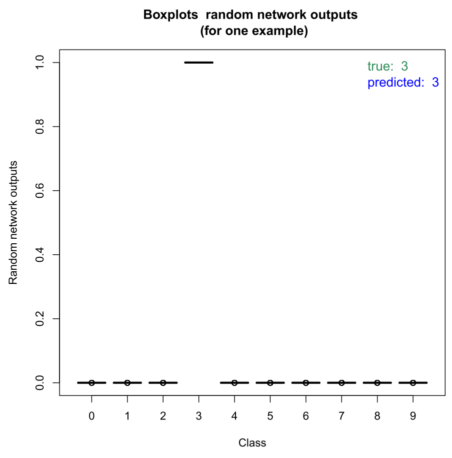


Fig. 3. Boxplots of 200 random network outputs for a representative correct classification result. Model with 100 neurons in the first fully connected layer.

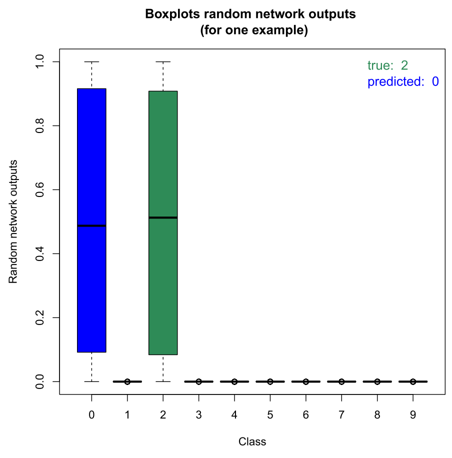


Fig. 4. Boxplots of 200 random network outputs for a representative incorrect classification result. Model with 100 neurons in the first fully connected layer.

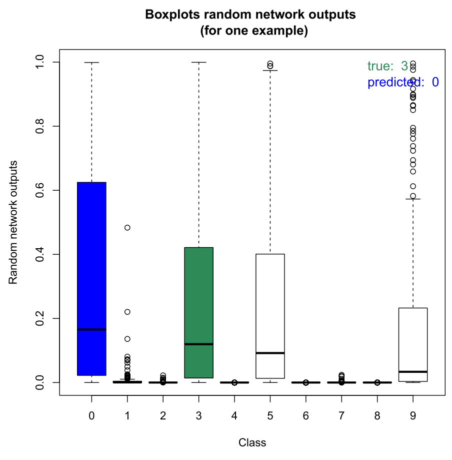
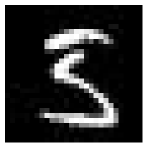


Fig. 5. Boxplots of 200 random network outputs for a representative incorrect classification result. Model with 100 neurons in the first fully connected layer.

intervals for the probability that an image shows an object of a given class can be estimated by computing multiple neural network outputs with weights sampled from the variational distribution. Figs. 3–5 show boxplots of, respectively, 200 random network outputs corresponding to three representative images



True: 5
Predicted: 3

Fig. 6. Test image where the model corresponding to the proposed approach was certain about its wrong prediction according to criterion (*ii*) (100 neurons in the first fully connected layer, $\alpha = 99.999\%$).

from the test data set. The network that was used to produce Figs. 3–5 is the one corresponding to our approach (Gauss cor.) with 100 neurons in the first fully connected layer. In Fig. 3, there are no boxes at all since the network is very sure about its correct prediction. This is the case for most of the correctly classified images. Figs. 4 and 5 reflect the common behavior of the net in the case of incorrect classification results. Either the network has difficulties to decide between two classes where one of the two is the true one or it is completely uncertain what class to predict. To quantify the usefulness of the prediction uncertainty of the Bayesian models, we consider two ways of measuring uncertainty. On the one hand, we estimate α credible intervals of the componentwise network outputs for all test images, and on the other hand, we estimate the posterior predictive distribution. The estimates are based on 200 random network outputs, respectively. For given α, a prediction result is considered as certain according to criterion (*i*) if the credible interval of the predicted class does not overlap with the intervals of the other classes and, moreover, a prediction is considered as certain according to criterion (*ii*) if its posterior probability is greater than or equal to α. Prediction results that are not classified as certain are considered as uncertain. Table III gives an overview of certain and uncertain prediction results, for different specifications of α. One can see that the models are, most of the time, certain about correct prediction results, while they are often uncertain about incorrect prediction results. As a consequence, the uncertainty information provided by each of the considered models can be viewed as useful. However, the models corresponding to the Gaussian approach without correlations (Gauss ind.) show, independent of the choice of α, always the highest number of certain but incorrect predictions. Also, specifying α near 100% does not allow for a shrinkage of this number to zero. In addition, the number of certain and correct prediction results is, most of the time, comparable to those of the other two methods. Thus, among the three considered approaches, this one appears to have the least useful uncertainty information if misclassifications with high confidence are critical. The other two approaches (Gauss. cor., Bernoulli) show similar fractions of certain and correct, or uncertain and incorrect classification results, but for different confidence levels. However, the Gaussian approach seems to reflect the specified confidence level α more reliable than the other one. According to criterion (*ii*), none of the certain prediction results is incorrect for $\alpha = 99\%$ in the case of the Bernoulli approach. However, since there are 8296 certain

TABLE III
OVERVIEW OF CERTAIN AND UNCERTAIN PREDICTION RESULTS

Model	Neurons	α	Prediction result	Certain (i)	Uncertain (i)	Certain (ii)	Uncertain (ii)
Gauss cor.	100	95%	correct	9668	262	9696	234
Gauss cor.	100	95%	wrong	14	56	13	57
Gauss cor.	100	99%	correct	9525	405	9524	406
Gauss cor.	100	99%	wrong	8	62	5	65
Gauss cor.	100	99.999%	correct	9372	558	7692	2238
Gauss cor.	100	99.999%	wrong	5	65	1	69
Gauss cor.	250	95%	correct	9587	352	9563	376
Gauss cor.	250	95%	wrong	8	53	8	53
Gauss cor.	250	99%	correct	9371	568	9133	806
Gauss cor.	250	99%	wrong	6	55	4	57
Gauss cor.	250	99.999%	correct	9112	827	4665	5274
Gauss cor.	250	99.999%	wrong	5	56	0	61
Gauss ind.	100	95%	correct	9735	160	9710	185
Gauss ind.	100	95%	wrong	35	70	24	81
Gauss ind.	100	99%	correct	9655	240	9519	376
Gauss ind.	100	99%	wrong	25	80	18	87
Gauss ind.	100	99.999%	correct	9590	305	8223	1672
Gauss ind.	100	99.999%	wrong	23	82	2	103
Gauss ind.	250	95%	correct	9661	239	9680	250
Gauss ind.	250	95%	wrong	19	81	17	83
Gauss ind.	250	99%	correct	9518	382	9503	397
Gauss ind.	250	99%	wrong	13	87	7	93
Gauss ind.	250	99.999%	correct	9394	506	8282	1618
Gauss ind.	250	99.999%	wrong	9	91	2	98
Bernoulli	100	95%	correct	9439	483	9243	679
Bernoulli	100	95%	wrong	7	71	3	75
Bernoulli	100	99%	correct	9049	873	8296	1626
Bernoulli	100	99%	wrong	2	76	0	78
Bernoulli	100	99.999%	correct	8558	1364	257	9665
Bernoulli	100	99.999%	wrong	2	76	0	78
Bernoulli	250	95%	correct	9545	377	9368	554
Bernoulli	250	95%	wrong	5	73	2	76
Bernoulli	250	99%	correct	9243	679	8551	1371
Bernoulli	250	99%	wrong	3	75	0	78
Bernoulli	250	99.999%	correct	8917	1005	814	9108
Bernoulli	250	99.999%	wrong	3	75	0	78

TABLE IV
LOG-LIKELIHOOD AND BRIER SCORE OF THE TESTING DATA SET

Model	Neurons	Log-likelihood	Brier score
Gauss cor.	100	-251.7366	0.01128257
Gauss cor.	250	-220.9156	0.01041743
Gauss ind.	100	-377.5755	0.0160266
Gauss ind.	250	-336.4502	0.01522011
Bernoulli	100	-302.7554	0.01338242
Bernoulli	250	-270.3255	0.01253694

TABLE V
TEST ERRORS OF THE TRAINED MODELS

Model	Test error
Frequentist	20.90%
Gauss cor.	20.32%
Gauss ind.	21.3%
Bernoulli	23.07%

TABLE VI
VARIATIONAL PARAMETERS ρ_j AND τ_j $(j = 1, \ldots, 4)$

Layer	τ_j	τ_{bj}	ρ_j	ρ_{bj}
Convolutional 1	0.04	0.05	-0.47	0.01
Convolutional 2	0.46	0.05	-0.49	-0.01
Convolutional 3	0.37	0.05	-0.41	0.01
Fully connected	0.14	0.05	-0.29	0.01

classifications, at least, some of them should be incorrect for $\alpha = 99\%$. Therefore, we consider the uncertainty information provided by our approach as the most useful. In Fig. 6, the test image can be found where the model corresponding to the proposed approach was certain about its wrong prediction according to criterion (*ii*) (100 neurons in the first fully connected layer, $\alpha = 99.999\%$). Finally, we evaluate the overall quality of the uncertainty information provided by the different approaches based on two measures. One the one hand, the log-likelihood and, on the other hand, the Brier score [47] are determined for the testing data set. The lower the Brier score is for a set of predictions, the better the

TABLE VII
OVERVIEW OF CERTAIN AND UNCERTAIN PREDICTION RESULTS

Model	α	Prediction result	Certain (i)	Uncertain (i)	Certain (ii)	Uncertain (ii)
Gauss cor.	95%	correct	5894	2074	4872	3096
Gauss cor.	95%	wrong	403	1629	148	1884
Gauss cor.	99%	correct	5281	2687	3592	4376
Gauss cor.	99%	wrong	205	1782	40	1992
Gauss cor.	99.999%	correct	4907	3061	668	7300
Gauss cor.	99.999%	wrong	179	1853	1	2031
Gauss ind.	95%	correct	7586	284	4611	3259
Gauss ind.	95%	wrong	1711	419	156	1974
Gauss ind.	99%	correct	7495	375	3154	4716
Gauss ind.	99%	wrong	1612	518	48	2082
Gauss ind.	99.999%	correct	7442	428	246	7624
Gauss ind.	99.999%	wrong	1531	599	0	2130
Bernoulli	95%	correct	4476	3217	3066	4627
Bernoulli	95%	wrong	181	2126	39	2268
Bernoulli	99%	correct	3611	4082	1641	6052
Bernoulli	99%	wrong	89	2218	12	2295
Bernoulli	99.999%	correct	3043	4650	45	7648
Bernoulli	99.999%	wrong	53	2254	0	2307

predictions are calibrated. For the calculation of these quantities, the already computed approximation of the posterior predictive distribution is used. The results can be found in Table IV. The proposed method achieves the best results.

B. CIFAR-10

In this section, based on the CIFAR-10 data set, the performance of the proposed method is evaluated and compared with other approaches (Frequentist, Gauss ind., Bernoulli). This data set consists of 60 000 RGB images in ten classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck). The complete data set is partitioned into a training data set that counts 50 000 images and a test set of 10 000 examples. Each image is of size 32×32. The architecture chosen for the performance evaluation is the one included under the name *CIFAR10_full* in the Caffe framework [45]. This CNN mainly consists of three convolutional layers followed by one fully connected layer. For further details, take a look at the model definition available in the Caffe framework.

Minibatch gradient descent with a batch size of 100 is chosen for optimization. Furthermore, a fixed learning rate policy with a learning rate of 0.001 is selected. The total amount of training iterations is given by 100 000 for the frequentist approach and by 40 000 for the Bayesian approaches. Penalization of the Euclidean norm with a penalization strength of 0.004 for the convolutional layers and a penalization strength of 1 for the fully connected layer is used for the regularization of the frequentist net. In our approach (Gauss cor.), we assign independent normal distributions with zero mean (see Section III-A) to the network parameters. The standard deviations of these distributions are specified as 1 for the convolutional layers and specified as 0.05 for the fully connected layer. Thus, the fully connected layer gets stronger regularized than the other ones, as in the frequentist case. Moreover, for each network layer, we set the parameters ν and κ described

TABLE VIII
LOG-LIKELIHOOD AND BRIER SCORE OF THE TESTING DATA SET

Model	Log-likelihood	Brier score
Gauss cor.	-6344.145	0.2927906
Gauss ind.	-6602.213	0.3068357
Bernoulli	-6617.044	0.3169693

in Section III-D to $1/(50\,000 \cdot 10)$ and 50, respectively. Thus, the penalization strength of the KL divergence is reduced by a factor of 10 due to the reasons described in Section III-D. For the Bayesian method without correlations (Gauss ind.), the prior and also the specification of ν are chosen identically to the specifications for the proposed approach (Gauss cor.). In the Bernoulli variational distribution, a hand-tuned dropping rate of 0.1 is used.

The test errors obtained from the trained models are given in Table V. Note that the predictions of the Bayesian models are based on 200 samples from the corresponding variational distributions per test image. Our approach achieves the lowest test error. In Table VI, the correlations ρ_j and the variance determining parameters τ_j of our Bayesian model can be found ($j = 1, \dots, 4$). The parameters can be interpreted, as in Section IV-A. The usefulness and quality of the prediction uncertainty information are evaluated, as in Section IV-A. A summary of the results is given by Tables VII and VIII. The new approach provides the most reliable uncertainty information overall.

V. CONCLUSION

We presented a Bayesian approach to deep learning that allows to learn correlations between network parameters while introducing only a few additional parameters to be optimized. In particular, we approximated the intractable posterior of the network parameters with a product of Gaussian distributions with tridiagonal covariance matrices. These distributions are defined in such a way that the variances are multiples of the squared expectation values and the correlations belonging to a given distribution are identical. The novel approach was evaluated on the basis of the popular benchmark data sets MNIST and CIFAR-10. Superior prediction accuracies compared with well-regularized frequentist models and also recent Bayesian approaches were obtained. Furthermore, we showed that useful uncertainty information about network predictions can be computed. Often the usefulness and quality of the provided uncertainty information were higher than the ones obtained from the comparison methods. Finally, network parameter uncertainties and dependencies could readily be interpreted per layer due to the fact that only a few additional parameters are required by our method.

Acknowledgment

This publication is one of the results of the project iDev40 (www.idev40.eu). The authors would like to thank the time spent and suggestions made by reviewers, which helped to improve the quality of this work.

References

[1] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain, "Neural probabilistic language models," in *Innovations in Machine Learning*, D. E. Holmes and L. C. Jain, Eds. Berlin, Germany: Springer, 2006, pp. 137–186.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst.*, 2012, vol. 25, pp. 1097–1105.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[5] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural Netw.*, vol. 3, no. 5, pp. 551–560, Jan. 1990.

[6] S. Liang and R. Srikant, "Why deep neural networks for function approximation?" 2016, *arXiv:1610.04161*. [Online]. Available: http://arxiv.org/abs/1610.04161

[7] K. M. Jozwik, N. Kriegeskorte, K. R. Storrs, and M. Mur, "Deep convolutional neural networks outperform feature-based but not categorical models in explaining object similarity judgments," *Frontiers Psychol.*, vol. 8, p. 1726, Oct. 2017.

[8] H. Greenspan, B. van Ginneken, and R. M. Summers, "Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique," *IEEE Trans. Med. Imag.*, vol. 35, no. 5, pp. 1153–1159, May 2016.

[9] V. Gulshan *et al.*, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *JAMA*, vol. 316, no. 22, pp. 2402–2410, Dec. 2016.

[10] K. Banerjee, T. Van Dinh, and L. Levkova, "Velocity estimation from monocular video for automotive applications using convolutional neural networks," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 373–378.

[11] J. B. Heaton, N. G. Polson, and J. H. Witte, "Deep learning for finance: Deep portfolios," *Appl. Stochastic Models Bus. Ind.*, vol. 33, no. 1, pp. 3–12, Jan. 2017.

[12] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Rel. Eng. Syst. Saf.*, vol. 172, pp. 1–11, Apr. 2018.

[13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Jun. 2014.

[14] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. 30th Int. Conf. Mach. Learn. (ICML)*, vol. 28, May 2013, pp. 1058–1066.

[15] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with Bernoulli approximate variational inference," 2015, *arXiv:1506.02158*. [Online]. Available: http://arxiv.org/abs/1506.02158

[16] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, Dept. Eng., Univ. Cambridge, Cambridge, MA, USA, 2016.

[17] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2575–2583.

[18] A. Roman, *Bayesian Deep Learning With Edward and a Trick Using Dropout*. London, U.K.: PyData, 2017.

[19] H. M. D. Kabir, A. Khosravi, M. A. Hosen, and S. Nahavandi, "Neural network-based uncertainty quantification: A survey of methodologies and applications," *IEEE Access*, vol. 6, pp. 36218–36234, 2018.

[20] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," *Proc. 33rd Int. Conf. Mach. Learn.*, 2015, pp. 1050–1059.

[21] S. K. Roy and P. Mula, "Solving matrix game with rough payoffs using genetic algorithm," *Oper. Res.*, vol. 16, no. 1, pp. 117–130, Apr. 2016.

[22] A. Bhaumik, S. K. Roy, and D.-F. Li, "Analysis of triangular intuitionistic fuzzy matrix games using robust ranking," *J. Intell. Fuzzy Syst.*, vol. 33, no. 1, pp. 327–336, Jun. 2017.

[23] S. K. Roy and A. Bhaumik, "Intelligent water management: A triangular Type-2 intuitionistic fuzzy matrix games approach," *Water Resour. Manage.*, vol. 32, no. 3, pp. 949–968, Feb. 2018.

[24] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6405–6416.

[25] H. M. D. Kabir, A. Khosravi, S. Nahavandi, and A. Kavousi-Fard, "Partial adversarial training for neural network-based uncertainty quantification," *IEEE Trans. Emerg. Topics Comput. Intell.*, early access, Sep. 5, 2019, doi: 10.1109/TETCI.2019.2936546.

[26] W. Buntine and A. S. Weigend, "Bayesian back-propagation," *Complex Syst.*, vol. 5, pp. 603–643, Jun. 1991.

[27] G. E. Hinton and D. van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proc. 6th Annu. Conf. Comput. Learn. Theory (COLT)*, 1993, pp. 5–13.

[28] J. Denker and Y. Lecun, "Transforming neural-net output levels to probability distributions," in *Proc. Adv. Neural Inf. Process. Syst.*, 1991, pp. 853–859.

[29] D. J. C. MacKay, "Probable networks and plausible predictions—A review of practical Bayesian methods for supervised neural networks," *Network: Comput. Neural Syst.*, vol. 6, pp. 469–505, May 1995.

[30] R. M. Neal, *Bayesian Learning for Neural Networks*, vol. 118. New York, NY, USA: Springer, 2012.

[31] A. Graves, "Practical variational inference for neural networks," in *Proc. NIPS*, 2011, pp. 2348–2356.

[32] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, vol. 37, 2015, pp. 1613–1622.

[33] C. Louizos and M. Welling, "Structured and efficient variational deep learning with matrix Gaussian posteriors," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn.*, vol. 48, 2016, pp. 1708–1716.

[34] S.-I. Amari, *Differential-Geometrical Methods in Statistics*. New York, NY, USA: Springer, 1985.

[35] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[36] J. M. Hernández-Lobato, Y. Li, M. Rowland, T. D. Bui, D. Hernández-Lobato, and R. E. Turner, "Black-box alpha divergence minimization," in *Proc. ICML*, vol. 48, 2016, pp. 1511–1520.

[37] Y. Li and Y. Gal, "Dropout inference in Bayesian neural networks with alpha-divergences," in *Proc. ICML*, 2017, pp. 2052–2061.

[38] K. Posch, J. Steinbrener, and J. Pilz, "Variational inference to measure model uncertainty in deep neural networks," 2019, *arXiv:1902.10189*. [Online]. Available: http://arxiv.org/abs/1902.10189

[39] L. Deng, "The MNIST database of handwritten digit images for machine learning research [Best of the Web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[40] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009, 2009.

[41] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5574–5584.

[42] C. M. Bishop, *Pattern Recognition Machine Learning Information Science Statistics*. New York, NY, USA: Springer, 2006.

[43] M. Anđelić and C. M. da Fonseca, "Sufficient conditions for positive definiteness of tridiagonal matrices revisited," *Positivity*, vol. 15, no. 1, pp. 155–159, Mar. 2011.
[44] J. R. Hershey and P. A. Olsen, "Approximating the kullback leibler divergence between Gaussian mixture models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, vol. 4, Apr. 2007, pp. 317–320.
[45] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," 2014, *arXiv:1408.5093*. [Online]. Available: http://arxiv.org/abs/1408.5093
[46] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Jun. 1998.
[47] G. W. Brier, "Verification of forecasts expressed in terms of probability," *Monthly Weather Rev.*, vol. 78, no. 1, pp. 1–3, 1950.

Konstantin Posch received the B.Sc. and M.Sc. degrees in technical mathematics from Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree in technical mathematics.

His research interests include Bayesian statistics, statistical machine learning, and big data analytics.

Juergen Pilz received the M.Sc., Ph.D., and D.Sc. degrees from the Freiberg University of Mining and Technology, Freiberg, Germany, in 1974, 1978, and 1988, respectively, all in mathematics and statistics.

He is currently a Professor of applied statistics with Alpen-Adria-Universität Klagenfurt (AAU), Klagenfurt, Austria, where he has been serving as the Head of the Department of Statistics since 2007. He has authored more than 150 publications in international journals and conference proceedings in the areas of Bayesian statistics, spatial statistics, environmental and industrial statistics, statistical quality control, and design of experiments. He is the author of seven books that appeared in internationally renowned publishing houses, such as Wiley, Springer, and Chapman and Hall and authored several book chapters.

Dr. Pilz is also an elected member of the International Statistical Institute (ISI) and the Institute of Mathematical Statistics (IMS). He is also on editorial boards of several international statistics journals.