Uncertainty quantification in neural network classifiers—A local linear approach

Magnus Malmström a,c,*, Isaac Skog b, Daniel Axehill a, Fredrik Gustafsson a

a Linköping University, Linköping, Sweden
b Uppsala University, Uppsala, Sweden
Swedish Defence Research Agency (FOI), Linköping, Sweden

A R T I C L E  I N F O

Article history:
Received 6 March 2023
Received in revised form 4 September 2023
Accepted 19 December 2023
Available online xxx

Keywords:
Neural networks
Uncertainty descriptions
Information and sensor fusion
Identification and model reduction

A B S T R A C T

Classifiers based on neural networks (NN) often lack a measure of uncertainty in the predicted class. We propose a method to estimate the probability mass function (PMF) of the different classes, as well as the covariance of the estimated PMF. First, a local linear approach is used during the training phase to recursively compute the covariance of the parameters in the NN. Secondly, in the classification phase, another local linear approach is used to propagate the covariance of the learned NN parameters to the uncertainty in the output of the last layer of the NN. This allows for an efficient Monte Carlo (MC) approach for: (i) estimating the PMF; (ii) calculating the covariance of the estimated PMF; and (iii) proper risk assessment and fusion of multiple classifiers. Two classical image classification tasks, i.e., MNIST, and CFAR10, are used to demonstrate the efficiency of the proposed method.

1. Introduction

In this paper, the problem of quantifying the uncertainty in the predictions from a neural network (NN) is studied. The uncertainty in the prediction stems from three different sources: errors caused by the optimization algorithm that is used to train the NN, errors in the data (aleatoric uncertainty), and errors in the model (epistemic uncertainty). In this paper, the focus is on uncertainty from the two latter sources.

In numerous applications, e.g., image recognition (Krizhevsky, Sutskever, & Hinton, 2012), and various control tasks (Karlsson & Hendeby, 2021; Li et al., 2017), NNs have shown high performance. Despite their high performance, the use of NNs in safety-critical applications is limited (Grigorescu, Trasnea, Cocias, & Macesanu, 2020; Paleyes, Urma, & Lawrence, 2020). It is partly a consequence of the fact that their predictions usually do not come with any measure of certainty of the prediction, which is crucial to have in a decision-making process to know to what degree the prediction can be trusted. In particular, this need was highlighted in the fatal Uber accident in 2018 where the lack of reliable classifications of surrounding objects played a role in the development of events that eventually led to the accident (NTSB, 2018). Moreover, the quantified measure of uncertainty can be used to detect and remove outliers in the data, and without knowledge about the uncertainty it is not possible to fuse the prediction from the NN with information from other sensors.

The problem to quantify the uncertainty in the prediction of NNs has lately gained increasing attention, and numerous methods to calculate the uncertainty have been suggested (D’Amour et al., 2020; Ghahramani, 2015; Lin, Clark, Trigoni, & Roberts, 2022; Ovadia et al., 2019; Patel & Waslander, 2022). For a survey of methods see Gawlikowski et al. (2023). The methods suggested in the literature can broadly be divided into one out of two categories. One category is based on creating an ensemble of predictions from which the uncertainty in the prediction is computed (Ayang & Berens, 2018; Gal & Ghahramani, 2016; Ilg et al., 2018; Lakshminarayanan, Pritzel, & Blundell, 2017; Maddox, Izmailov, Garipov, Vetrov, & Wilson, 2019; Osawa et al., 2019; Teye, Azizpour, & Smith, 2018). In the other category, the NN structure is extended and the NN is trained to learn its own uncertainty (Blundell, Cornebise, Kavukcuoglu, & Wierstra, 2015; Charpentier, Züger, & Günemann, 2020; Gustafsson, Danelljan, Bhat, & Schön, 2020; Izmailov, Nicholas, Lof, & Wilson, 2021; Kendall & Gal, 2017). Concerning the first category, it has for example been suggested to create an ensemble by training multiple NNs, from whose predictions the uncertainty...
is computed by Lakshminarayanan et al. (2017). Since training a single \( \text{nN} \) is often computationally expensive, this method has high computational complexity. In practice, it is only feasible from a computational perspective to create small ensembles. To decrease the computational complexity, it was in Gal and Ghahramani (2016) and Teye et al. (2018) suggested to use already existing regularization techniques to sample values of the parameters of the \( \text{nN} \) to create these ensembles. Another method to create ensembles is by sampling values of the parameters during the last part of the training phase (Maddox et al., 2019; Osawa et al., 2019). So-called test-time data augmentation methods have also been suggested to do perturbation on the test data to create an ensemble of predictions (Ayhan & Berens, 2018). Even though the methods in Ayhan and Berens (2018), Gal and Ghahramani (2016), Maddox et al. (2019), Osawa et al. (2019) and Teye et al. (2018) do not need multiple models to be trained they require multiple forward passes. Furthermore, they require specially tailored training algorithms and carefully constructed structures of the models.

Another limitation of methods relying on creating ensembles is that they have trouble representing the uncertainty caused by the bias in the prediction from a model mismatch. The bias can be caused by an insufficiently flexible model, which could be a result of too high regularization or too low model order. The problem can be solved by \( \text{nnN} \) from the second category, i.e., where the structure of the \( \text{nN} \) is extended such that it learns its own uncertainty in the prediction. However, this requires a more intricate \( \text{nN} \) structure with tailored loss functions (Blundell et al., 2015; Charpentier et al., 2020; Gustafsson et al., 2020; Kendall & Gal, 2017). As a consequence, the training becomes more complex and computationally expensive. It also makes the methods sensitive to errors caused by the training algorithm, which are not possible to learn. Furthermore, there is also a need for more data to train complex model structures.

In this paper, we address the two limitations of the aforementioned methods using classical local approximations from the area of system identification (Ljung, 1999) based on a linearization of the model. Where the system identification methodology works well for static systems which \( \text{nN} \) are an example of Avant and Morgansen (2023) and Shen, Varshney, and Zhu (2014). We will refer to this as the delta method (Liero & Zwanzig, 2011; Malmström, 2021; Malmström, Skog, Axehill, & Gustafsson, 2022). For regression tasks, the delta method has previously been used to capture the uncertainty in the prediction of \( \text{nN} \), see e.g., Deng, Zhou, and Zhu (2022), Hwang and Ding (1997), Immer, Korzepa, and Bauer (2021), Liero and Zwanzig (2011) and Malmström (2021), and extended to classification tasks in Malmström et al. (2022).

2. Problem formulation and contributions

Consider the problem of learning a classifier from the training data set

\[
\mathcal{T} \triangleq \{(y_n, x_n)\}_{n=1}^N
\]

(1)

Here \( y_n \in \{1, \ldots, M\} \) is the class labels and \( x_n \in \mathbb{R}^n \) is the input data of size \( n_n \), e.g., pixels in an image. From a statistical point of view, the meaning of the classifier can be seen as a system identification problem where a model \( f(x; \theta) \) that predicts the conditional probability mass function (PMF) \( \hat{p}(y|x) \) of a categorical distribution, are to be modeled. That is, the probability for \( y = m \) given the input \( x \) is modeled as

\[
p(y = m|x; \theta) = f_m(x; \theta), \quad m = 1, \ldots, M.
\]

(2)

Here \( \theta \in \mathbb{R}^n \) denotes the \( n_{\theta} \)-dimensional parameter vector that parameterizes the model. Furthermore, the subscript \( m \) denotes the \( m \)th element of the vector-valued output of the function. To ensure that the model \( f(x; \theta) \) fulfills the properties associated with a PMF, i.e., \( f_m(x; \theta) \geq 0 \ \forall m \) and \( \sum_m f_m(x; \theta) = 1 \), it is typically structured as

\[
f(x; \theta) = \text{softmax}(g(x; \theta)),
\]

\[
\text{softmax}(z) \triangleq \frac{1}{\sum_{m=1}^M e^{z_m}}
\]

(3a)

and \( g(x; \theta) \) describes the underlying model of the classifier. For example, if \( g(x; \theta) \) is given by an \( \text{nnN} \) the parameters \( \theta \) consist of all the weights and biases in the \( \text{nN} \). In the case \( g_m(x; \theta) = \theta^\top \phi_m(x) \), where \( \phi_m(x) \) denotes, a possible nonlinear, transformation of the input \( x \), then the model in (3a) becomes a standard multinomial logistic regression model (Baggio, Carè, Scampicchio, & Pillonetto, 2022; Lindholm, Wahlström, Lindsten, & Schön, 2022). Furthermore, if the transformation \( \phi_m(x) \) is chosen randomly, the model becomes similar to the one used in extreme learning machine classifiers (Huang, Zhu, & Siew, 2006).

2.1. Parameter estimation

For most \( \text{nN} \) the number of model parameters \( n_{\theta} > N \) and the model parameters \( \theta \) cannot be uniquely identified from the training data \( \mathcal{T} \) without some regularization or prior information regarding the parameters. Let \( p(\theta) \) denote the prior for the model parameters. The maximum a posteriori estimate of the model parameters is then given by

\[
\hat{\theta}_N = \arg\max_{\theta} p(\theta|\mathcal{T}) = \arg\max_{\theta} L_N(\theta) + \ln p(\theta),
\]

(4)

where \( p(\theta|\mathcal{T}) \) denotes the a posteriori distribution of the parameters and

\[
L_N(\theta) = \sum_{n=1}^N \ln f_m(x_n; \theta)
\]

(5)

denotes the cross-entropy likelihood function (Lindholm et al., 2022). Here \( y_n \) is used as an index operator for the subscript \( m \) of \( f_m(x_n; \theta) \).

2.2. Prediction and classification

Once the classifier has been learned, i.e., a parameter estimate \( \hat{\theta}_N \) has been computed, then for a new input data point \( x^* \) the PMF can be predicted as

\[
\hat{p}(y^* = m|x^*; \hat{\theta}_N) = f_m(x^*; \hat{\theta}_N), \quad m = 1, \ldots, M
\]

(6)

and the most likely class can be found as \( \hat{y}^* = \arg\max_m f_m(x^*; \hat{\theta}_N) \).

Note that, the full PMF estimate \( f(x^*; \hat{\theta}_N) \) is needed both for temporal fusion using several inputs from the same class and fusion over different classifiers. Furthermore, even small probabilities can pose a large risk, e.g., there might be a pedestrian in front of a car even if another harmless object is more likely according to the classifier. Hence, it is important that the prediction \( \hat{p}(y^* = m|x^*; \hat{\theta}_N) \) is accurate. However, it is well known that due to among other things, uncertainties in the parameter estimates \( \hat{\theta}_N \) the disagreement between true and estimated PMF may be significant. Therefore, methods to calibrate the prediction \( \hat{p}(y^* = m|x^*; \hat{\theta}_N) \) such that it better matches \( p(y^* = m|x^*) \) has been developed.

2.3. Temperature scaling

One of the most commonly used methods to calibrate the predicted PMF is called temperature scaling (Guo, Pleiss, Sun, & Weinberger, 2017). In temperature scaling, \( g(x; \theta) \) is scaled
by a scalar quantity $T$ before the normalization by the softmax operator. With a slight abuse of notation, introduce
\[ f(x^i; \hat{\theta}_n, T) = \text{softmax} \left( g(x^i; \hat{\theta}_n) / T \right). \] (7)

Using the temperature scaling parameter $T$, the variations between the components (classes) in the predicted PMF can be enhanced or reduced. When $T \to 0$, then $f(x^i; \hat{\theta}_n, T) \to \delta_i$, \textit{where} $\delta_i$ \textit{denotes} the $i$th standard basis vector, thereby indicating that input $x^i$ with total certainty belongs to class $i$. Similarly, when $T \to \infty$, then $f(x^i; \hat{\theta}_n, T) \to 1 / M \forall n$, thereby indicating that input $x^i$ is equally probable to belong to any of the classes. Noteworthy is that the temperature scaling is typically done after the parameters $\theta$ have been estimated, where it is estimated using validation data. For notational brevity, the dependency on the temperature scaling parameter $T$ will only be explicitly stated when temperature scaling is considered.

2.4. Marginalization of parameter uncertainties

A more theoretically sound approach to take the uncertainties in the parameter estimate into account is via marginalization of the PMF with respect to the parameter distribution. That is, an estimate of the PMF and its covariance are calculated as
\[ f(x^i | T) \triangleq \int f(x^i; \theta) p(\theta | T) d\theta \]
\[ p^f \triangleq \int f(x^i; \theta) - f(x^i | T) \rangle \text{p}(\theta | T) d\theta \] (8a)

From hereon, $(x;\rangle^T$ is used as a shorthand notation for $xx^T$. The integral in (8a) is generally intractable, but can be approximated by Monte Carlo (MC) sampling as
\[ \theta^{(k)} \sim \text{p}(\theta | T), \quad k = 1, 2, \ldots, K., \]
\[ \hat{f}(x^i | T) = \frac{1}{K} \sum_{k=1}^{K} f(x^i; \theta^{(k)}) \]
\[ \hat{p}^f = \frac{1}{K} \sum_{k=1}^{K} (f(x^i; \theta^{(k)}) - \hat{f}(x^i | T)) \rangle^T. \] (8c)

Here $K$ denotes the number of samples used in the MC sampling.

2.5. Challenges and contributions

To realize the MC scheme in (9) the posterior parameter distribution $p(\theta | T)$ must be computed and samples drawn from this high-dimensional distribution. Our contributions are: (i) a local linearization approach that leads to a recursive algorithm of low complexity to compute an approximation of $p(\theta | T)$ during the training phase; (ii) a second local linearization approach to reduce the sampling space from $n_u$ to $M$-dimensional space in the prediction phase; and as a by-product (iii) a low-complexity method for risk assessment and information fusion.

3. Posterior parameter distribution

Next, a local linear approach to compute an approximation of $p(\theta | T)$ during the training phase is presented.

3.1. Laplace approximation

Assume the prior distribution for the model parameters to be normal distributed as $p(\theta) = \mathcal{N}(\theta; 0, P_0)$, i.e., $P$ regularization is used. Then a Laplacian approximation of the posterior distribution $p(\theta | T)$ yields that (Bishop, 2006)
\[ p(\theta | T) \approx \mathcal{N}(\theta; \hat{\theta}_n, P_N^\theta), \]
(10a)

\[ p_{N}^\theta = \left( -\frac{\partial^2 \ln p^f(\theta)}{\partial \theta^2} \bigg|_{\theta = \hat{\theta}_n} + P_0^{-1} \right)^{-1}. \] (10b)

That is, the prior distribution is approximated by a normal distribution with a mean located at the maximum a posteriori estimate and a covariance dependent upon the shape of the likelihood function in the vicinity of the estimate. The accuracy of the approximation will depend upon the amount of information in the training data $T$. A weakness with the Laplacian approximation is that it is local and assumes $\hat{\theta}_n$ to be a local minimum and can hence miss global trend in the posterior distribution (Bishop, 2006).

3.2. Asymptotic distribution

According to Bernstein–von Mises theorem (Johnstone, 2010), if the true model belongs to the considered model set, the maximum a posteriori estimate $\hat{\theta}$ converges in distribution to
\[ \hat{\theta}_n \xrightarrow{d} \mathcal{N}(\theta_*; \theta, I_u^{-1}), \] (11)
where the information in the training data $T$ tends to infinity. Here, $\theta_*$ denotes the true parameters and
\[ I_u \triangleq -E \left[ \frac{\partial^2 \ln p(\theta)}{\partial \theta^2} \right], \] (12)

is the Fisher information matrix. Given the likelihood function in (5) the Fisher matrix becomes
\[ I_u \simeq \sum_{n=1}^{N} \sum_{m=1}^{M} \eta_{m,n} \frac{\partial g_m(x_n; \theta)}{\partial \theta} \left( \frac{\partial g_m(x_n; \theta)}{\partial \theta} \right)^\top, \]
(13a)
\[ \eta_{m,n} \triangleq f_m(x_n; \theta | 1 - f_m(x_n; \theta)). \] (13b)

See derivations in the Appendix.

3.3. Recursive computation of covariance

To compute the parameter covariance $P_N^\theta$ defined by (10b), the Hessian matrix of the log-likelihood ($\lambda$) must be calculated and then inverted. This has a complexity of $O(NMn_u^2 + n_u^3)$, which for large $n_u$ in combination with large $N$ can become intractable. However, by approximating the Hessian matrix of the $\lambda$ with the Fisher information matrix using (13), the computation can be done recursively and with a complexity of $O(NMn_u^2 + NM^3)$ (Malmström et al., 2022). Note that $I_u$ in (13) can be written in a quadratic form. As a result the recursive update can be given as
\[ K_n = P_n^\theta U_n \left( I_M + U_n^\top P_n^\theta U_n \right)^{-1}, \]
\[ P_{n+1} = P_n^\theta - K_n U_n P_n^\theta, \]
\[ U_n = \left[ u_{1,n}, \ldots, u_{M,n} \right] \in \mathbb{R}^{n_u \times M}, \]
\[ u_{m,n} \triangleq \sqrt{\eta_{m,n} - \frac{\partial g_m(x_n; \theta)}{\partial \theta} \bigg|_{\theta = \hat{\theta}_n}. \]
(14d)

where $I_r$ denotes the identity matrix of size $r$. Here $P_n^\theta$ is the parameter covariance for $n$ measurements, which is initialized as $P_0^\theta = P_0$. To compute $u_{m,n} \in \mathbb{R}^{n_u}$ only the gradient of the $\lambda$ in (5) is required, which is nevertheless needed for the estimation of $\theta$.

3.4. Approximating the covariance

An NN often has millions of parameters which might result in the amount of data needed to store $P_n^\theta$ being larger than the available memory capacity. A common approach to handle this is
to approximate \( P_N^\theta \) as a block-diagonal matrix (Martens & Grosse, 2015). Another common approach is to use the approximation
\[
P_N^\theta \approx \begin{pmatrix} P_{N_r}^\theta & 0 \\ 0 & 0 \end{pmatrix},
\]
where \( P_N^\theta \) denotes the covariance of the estimated parameters \( \theta \) corresponding to the weights and biases of the \( r \) last layers in the NN (Krisstiainen, Hein, & Hennig, 2020; Malmström et al., 2022). Depending on the number of included layers, this approximation might be more or less accurate. To compensate for the approximation error when doing the marginalization in (8), a scaling of \( P_N^\theta \) with factor \( T_r \geq 1 \) can be introduced. The scaling can be estimated from validation data in a similar manner to the temperature scaling \( T \) in Section 2.3.

4. Efficient MC sampling

With access to the parameter covariance, one can propagate the uncertainty in the parameters to uncertainty in the prediction with the delta method using the principle of marginalization. Plugging in the approximate Gaussian distribution \((11)\) into \((8a)\) with the delta method using the principle of marginalization. 4. Efficient MC sampling

4.1. Marginalization using the delta method

The delta method, see e.g., Liero and Zwanzig (2011) and Malmström (2021), relies on linearization of the nonlinear model \( g(x, \theta) \) and provides a remedy to the problem of sampling from the high-dimensional Gaussian distribution. The idea is to project the uncertainty in the parameters to uncertainty in the prediction before the softmax normalization (3b), thereby drastically reducing the dimension of the distribution that must be sampled. Using the delta method, the uncertainty in the parameters can be propagated to the prediction before the softmax normalization as
\[
\hat{\theta}^{(k)} \sim N(\hat{\theta}_N; P_N^\theta), \quad k = 1, 2, \ldots, K.
\]
This is a feasible solution to the problem, but it comes with a high computational cost since it requires drawing MC samples from a high-dimensional Gaussian distribution and evaluating the whole network.

4.1.1. Marginalization using the delta method

The delta method, see e.g., Liero and Zwanzig (2011) and Malmström (2021), relies on linearization of the nonlinear model \( g(x, \theta) \) and provides a remedy to the problem of sampling from the high-dimensional Gaussian distribution. The idea is to project the uncertainty in the parameters to uncertainty in the prediction before the softmax normalization (3b), thereby drastically reducing the dimension of the distribution that must be sampled. Using the delta method, the uncertainty in the parameters can be propagated to the prediction before the softmax normalization as
\[
p(g(x^*; \theta)|T) \approx N\left( g(x^*; \theta); \hat{g}_N, P_N^p \right), \quad \hat{g}_N = E[g(x^*; \theta)] \approx g(x^*; \hat{\theta}_N),
\]
\[
P_N^p = \text{Cov}[g(x^*; \theta)] \approx \left( \frac{\partial}{\partial \theta} g(x^*; \theta) \right) P_N^\theta \left( \frac{\partial}{\partial \theta} g(x^*; \theta) \right)^\top.
\]
Using this Gaussian approximation of the parameter distribution, the MC approximation of the marginalization integral becomes
\[
g^{(k)}(x^*) \sim N\left( g(x^*; \theta); \hat{g}_N, P_N^p \right), \quad k = 1, 2, \ldots, K
\]
\[
f^{(k)}(x^*) = \text{softmax}(g^{(k)}(x^*)),
\]
\[
\hat{f}^i(x^*|T) = \frac{1}{K} \sum_{k=1}^K f^{(k)}(x^*),
\]
\[
\hat{p}^i = \frac{1}{K} \sum_{k=1}^K \left( f^{(k)}(x^*) - \hat{f}^i(x^*|T) \right)(\cdot)^\top.
\]

To summarize, the main idea of the delta method is linearization performed in two steps. First, the parameter uncertainty is computed using (11), and second, the uncertainty is propagated to the output of the model by (17). Here, the delta method is a local linear approach that gives a linear approximation of a nonlinear model.

4.2. Fusion

Suppose there is a set of independent classifiers, each one providing a marginal distribution \( N_i \left( \hat{g}_N,c_i; \hat{g}_N,c_i, P_{c_i}^N \right) \), \( c = 1, \ldots, C \). Then the predictions (before the softmax normalization) from these classifiers can be fused as follows (Gustafsson, 2018)
\[
p_N^\epsilon = \left( \sum_{c=1}^C (P_{c_i}^N)^{-1} \right)^{-1},
\]
\[
\hat{g}_N = P_N^\epsilon \sum_{c=1}^C (P_{c_i}^N)^{-1} \hat{g}_N,c.
\]
If a single classifier is used to classify a set of inputs \( x^*_c, c = 1, \ldots, C \), known to belong to the same class \( y^* \), then these predictions can be fused as follows
\[
p_N^\epsilon = (H^\top R^{-1} H)^{-1},
\]
\[
\hat{g}_N = P_N^\epsilon H^\top R^{-1} z,
\]
\[
z = \left[ \hat{g}_{N,1}, \ldots, \hat{g}_{N,C} \right]^\top, \quad H = \left[ I_{n_0} \ldots I_{n_0} \right]^\top,
\]
\[
[R]_{ij} = \left( \frac{\partial}{\partial \theta} g(x^*_c; \theta)^\top \right)_{|_{\theta = \theta_N}}, \quad i, j = 1, \ldots, C.
\]
where \( z \in \mathbb{R}^{Cn_0}, H \in \mathbb{R}^{Cn_0 	imes n_0} \) and the block \( [R]_{ij} \in \mathbb{R}^{n_0 \times n_0} \).

After fusion, the MC sampling in (18) can be applied as before to compute the PMF estimate.

4.3. Risk assessment

Risk assessment can be defined as the probability \( r_m \) that \( p(y^*_i = m|x^*_i) > \gamma_m \). The probability \( r_m \) can be estimated from the identified model \( f_m(x^*_i|T) \) as follows
\[
r_m = \text{Pr}(f_m(x^*_i|T) > \gamma_m) \approx \frac{1}{K} \sum_{k=1}^K \mathbf{1}(f^{(k)}_m(x^*_i) > \gamma_m).
\]

5. Validation

Suppose now we have a validation data set \( \mathcal{V} = \{ y^*_n, x^*_n \}_{n=1}^N \). In this section we will investigate metrics how to validate the estimated PMF \( f(x^*_i|T) \) obtained from (18). The inherent difficulty is that the validation data, just as the training data, consists of inputs and class labels, not the actual PMF. Indeed, there is a lack of unified qualitative evaluation metrics (Gawlikowski et al., 2023). That being said, some of the most commonly used metrics are classification accuracy, IL, Brier score, and expected calibration error (ECE) (Wójcik, Grela, Śmieja, Misztal, & Tabor, 2022). Both the negative IL and the Brier score are proper scoring rules, meaning that they emphasize careful and honest assessment of the uncertainty, and are minimized for the true probability vector (Gneiting & Raftery, 2007). However, neither of them is a measure of the calibration, i.e., reliability of the estimated PMF. Out of these metrics, only ECE considers the calibration. Hence, here ECE is the most important metric when evaluating a method used to measure the uncertainty (Guo et al., 2017; Vaikcnavicius et al., 2019). To evaluate a method that quantifies uncertainty in the prediction, one can also measure its capability to distinguish between in- and out-of-distribution (iD and oD) inputs, i.e., find outliers.
5.1. Brier score, accuracy, and reliability diagram

The Brier score (Gneiting & Raftery, 2007) corresponds to the least squares fit
\[
\frac{1}{N_B} \sum_{n=1}^{N_B} \sum_{m=1}^{M} (\delta_{m,y_n} - \hat{p}(y_n^m = m|x_n^m))^2,
\]
(22)
where \( \delta_{i,j} \) denotes the Kronecker delta function. Furthermore, \( \hat{p}(y_n^m = m|x_n^m) \) denotes a generic PMF estimate.

Accuracy and reliability diagrams are calculated as follows. Calculate the \( B_j \) histogram defined as
\[
B_j = \left\{ n : \frac{j-1}{j} \leq \max_m \hat{p}(y_n^m = m|x_n^m) < \frac{j}{j} \right\}
\]
(23)
from the validation data. For a perfect classifier \( B_j = \emptyset \) for \( j < J \). For a classifier that is just guessing, all sets are of equal size, i.e., \( |B_j| = |B_j| \vee j \). Note that \( \max_m \hat{p}(y_n^m = m|x_n^m) \geq 1/M \), so the first bins will be empty if \( j > M \). The accuracy of the classifier is calculated by comparing the size of each set with the actual classification performance within the set. That is,
\[
\text{acc}(B_j) = \frac{1}{|B_j|} \sum_{n \in B_j} \delta(y_n^m = y_n^m)
\]
(24)
where \( \hat{y}_n^m = \arg \max_m \hat{p}(y_n^m = m|x_n^m) \). A reliability diagram is a plot of the accuracy versus the confidence, i.e., the predicted probability frequency. A classifier is said to be calibrated if the slope of the bins is close to one, i.e., when \( \text{acc}(B_j) = (j - 0.5)/J \) (Guo et al., 2017).

5.2. Confidence and expected calibration error

Instead of certainty, from hereon the standard, and equivalent, notion of confidence will be used (Guo et al., 2017; Vaidenavicius et al., 2019). The mean confidence in a set is denoted \( \text{conf}(B_j) \) and is defined as
\[
\text{conf}(B_j) = \frac{1}{|B_j|} \sum_{n \in B_j} \max_m \hat{p}(y_n^m = m|x_n^m),
\]
(25)
This is a measure of how much the classifier trusts its estimated class labels. In contrast to the accuracy it does not depend on the annotated class labels \( y_n \). Comparing accuracy to confidence gives the ece, defined as
\[
\text{ece} = \sum_j \frac{|B_j|}{J} |\text{acc}(B_j) - \text{conf}(B_j)|.
\]
(26)
A small value indicates that the weight is a good measure of the actual performance.

5.3. Out-of-distribution detection

To evaluate the capability of a method to find ood samples, the so-called area under the receiver operating characteristic (AUROC) curve and area under the precision-recall curve (AUPR) is often used (Goodfellow, Bengio, & Courville, 2016). Both the AUROC and the AUPR are between zero and one, and for a perfect detector, the value is one. Intuitively, AUROC is a measure of the proportion of detections, while AUPR measures the proportions of detections that are correct. Another approach to evaluate the capability of ood detection is to study the difference in entropy between \( \text{id} \) and ood inputs \( \Delta S = S_{\text{id}} - S_{\text{ood}} \) where the entropy \( S \) is defined as
\[
S = -\sum_{m=1}^{M} \hat{p}(y_n^m = m|x_n^m) \ln \left( \hat{p}(y_n^m = m|x_n^m) \right).
\]
(27)

6. Experiment study

To illustrate the application of the proposed method to quantify uncertainty in the prediction, two datasets were used. First, an nn was trained using the \( \text{mnist} \) dataset (LeCun, Cortes, & Burges, 1998) to classify images of handwritten digits. Second, an nn was trained on the cfar10 dataset (Krizhevsky, 2009) to classify images of ten different objects including e.g., cars, cats, and aircraft. For ood detection \( \text{mnist} \) was used as \( \text{id} \) and Fashion \( \text{mnist} \) (Fmnist) (Xiao, Rasul, & Vollgraf, 2017) was used as ood.

6.1. Classification setup

For the \( \text{mnist} \) dataset, a five-layer nn with fully-connected nodes was used. For the cfar10 dataset, a LeNet5-inspired structure was used with six convolutional layers followed by four fully connected layers. However, for both datasets, the three last layers were chosen to have the same structure, fully connected with 100, 40, and 10 hidden nodes. To decrease the size of the parameter covariance matrix used by the delta method, as described in Section 3.4, the first part of the nn was assumed fixed and used to create high-level features. Since the structure of the later layers was chosen identically, both models had \( n_p = 4450 \) parameters. To estimate the model parameters \( \theta \) of the nn, the adam optimizer (Kingma & Ba, 2015) was used with standard settings. Three and ten epochs were used to train the \( \text{mnist} \) dataset and cfar10 dataset, respectively. Here, \( l^2 \) regularization of \( 10^{-4} \) were used, hence the prior \( P_0 = 10^{-4} I_{n_p} \). When computing \( P_\theta \), the choice of \( P_0 \) can be neglected since it is dominated by the Fisher information matrix \( I_\theta \).

6.2. Illustration of the uncertainties in the predictions

The low-dimensional space of the output from \( g(x_\theta^m; \hat{\theta}_n) \) is particularly interesting to study when trying to understand how the uncertainty in the parameter estimate \( \hat{\theta}_n \) affects the classification. Even if the parameter covariance \( P_\theta^m \) is constant and only depends on the training data, the covariance \( P_\theta^\theta \) depends on the input \( x_n \). Fig. 1 illustrates this via an example where we concentrate our study on the decision between just a subset of the number of classes in the \( \text{mnist} \) dataset, even though the final decision is over all classes. More generally, for an input \( x_n \) that is located in a dense region in the space of the training data, the covariance \( P_\theta^\theta \) is small, but for an input \( x_n \) that is very far from the training data in some norm, the covariance \( P_\theta^\theta \) can be quite large. This indicates that the parameter estimate is quite sensitive in some
directions. Even though the estimate of the pmf looks similar, by studying the unnormalized prediction \( \hat{p}(x; \hat{\theta}_h) \) it is clear that the prediction in the top has a higher uncertainty compared to the bottom one.

6.3. Results on quantifying the uncertainty

Six different methods to quantify the uncertainty in the classification, i.e., to estimate \( \hat{p}(y = m|x; \hat{\theta}_h) \), were evaluated. These are: (i) Standard method, i.e., \( \hat{p}(y = m|x) = f_m(x; \hat{\theta}_h) \); (ii) Temp. scaling, i.e., \( \hat{p}(y = m|x) = f_{\text{mc}}(x; \hat{\theta}_h, T) \), Guo et al. (2017); (iii) Deep ensemble, i.e., \( \hat{p}(y = m|x) \) is estimated using the ensemble method in Lakshminarayanan et al. (2017); number of trained nns are 50 for \( \text{mnist} \) and 10 for \( \text{cfar10} \); (iv) \( \text{mc-dropout} \), i.e., \( \hat{p}(y = m|x) \) is estimated using the ensemble method in Gal and Ghahramani (2016); 50 samples of the parameters are used to create the ensemble; (v) Proposed method, i.e., \( \hat{p}(y = m|x) = \hat{f}_m(x; \hat{T}_c, T) \); (vi) Proposed method with scaled covariance, i.e., \( \hat{p}(y = m|x) = \hat{f}_m(x; \hat{T}_c, T) \), but with the covariance \( \hat{P}^T \) in (17) scaled with a factor \( T_c \).

The parameters \( T \) and \( T_c \) are estimated such that \( \text{ece} \) is minimized on validation data. In Table 1, the accuracy, \( \text{ll} \), Brier score, and \( \text{ece} \) are shown for six different methods evaluated both using the \( \text{mnist} \) and \( \text{cfar10} \) datasets. Table 1 shows that the proposed method attains the lowest \( \text{ece} \) for both datasets, while still having reasonably good performance in terms of accuracy, \( \text{ll} \), and Brier score. Neither computing the uncertainty in the prediction using the softmax (i), deep ensembles (iii), \( \text{mc-dropout} \) (iv), nor the proposed method without scaled covariance (v) gives calibrated estimates of the uncertainty. To get well-calibrated estimates of the uncertainty either the proposed method with scaled covariance (vi) or temperature scaling (ii) should be used. However, increasing the scaling factor decreases the \( \text{ll} \). Hence, there is a trade-off between high \( \text{ll} \) and low \( \text{ece} \). In Fig. 2, the reliability diagram for the calibrated and non-calibrated versions of the proposed method is shown. Here one can see that the parameter \( T_c \) is necessary in order to obtain a calibrated uncertainty. For out detection, in Table 2 and Fig. 3, one can see that the proposed method is superior compared to temperature scaling (which is the only other calibrated method), while having similar performance to a deep ensemble whose performance is the best. However, recall that deep ensemble requires training of 50 models, hence, having significantly higher computational complexity compared to the proposed method.

On a standard laptop (Intel core i7), computing (17c) takes 930 s where the sampling in (18) is in comparison negligible (100 samples in 5 s). This is compared to sampling the parameters and then performing a forward pass per sample (such as in \( \text{mc-dropout} \) and deep ensemble) where only the forward pass takes 50 s. This is without considering that methods such as deep ensemble require training multiple nn.

### 7. Summary and conclusion

A method to estimate the uncertainty in classification performed by an nn has been proposed. The method also enables information fusion in applications where predictions from nns are used as well as statistical risk assessment. The proposed method is based on a local linear approach and consists of two steps. In the first step, an approximation of the posterior distribution of the estimated nn parameters is calculated. This is done using a Laplacian approximation where the covariance of the parameters is calculated recursively using the structure of the Fisher information matrix. In the second step, an estimate of the pmf is calculated where the effect of the uncertainty in the estimated parameters is considered using marginalization over the posterior distribution of the parameter estimate. This is done.
by propagating the uncertainty in the estimated parameters to the uncertainty in the output of the last layer in the nn using a second local linear approach. The uncertainty in the output of the last layer is approximated as a Gaussian distribution of the same dimension as the number of classes. The pmf its covariance are then calculated via mc sampling, where samples are drawn from this low-dimensional distribution.

The proposed method has been evaluated on two classical classification datasets; mnist and cifar10. Nns with standard architectures were used. To handle a large number of parameters in these nn architectures, only the parameters of the last layers were considered in the uncertainty computations. The results, in terms of ece, show that the proposed method in its standard form yielded a similar performance as standard methods which do not take the uncertainty in the parameters into account. However, when using a rescaled parameter covariance matrix, used to compensate for the fact that only the uncertainty from the parameters in the last layers was considered, a significant reduction in the ece was observed. This illustrates that the proposed method works well. However, that more advanced low-rank methods to approximate the parameter covariance are then calculated via mc sampling, where samples are drawn from this low-dimensional distribution.

The last approximative equality follows from that $E[\delta_{m,y}] = p(y_m = y_n) = \delta_{m,y}$ and that $f_m(x_n, \theta)$ is an unbiased estimate of $p(y_m = y_n)$ when the information in the training data tends to infinity.

### References


Magnus Malmström received his B.Sc. and M.Sc. in Applied Physics and Electrical Engineering from KTH Royal Institute of Technology, Sweden in 2003 and 2005, respectively. In 2010, he received a Ph.D. degree in Signal Processing with a thesis on low-cost navigation systems. In 2009, he spent 5 months with the Mobile Multi-Sensor System research team, University of Calgary, Canada, as a visiting scholar, and in 2011 he spent 4 months at the Indian Institute of Science (IISc), Bangalore, India, as a visiting scholar. Between 2016 and 2017 he was a researcher at KTH Royal Institute of Technology, Stockholm, Sweden. Between 2017 and 2022 he was an associate professor at Linköping University, Sweden. Currently, he is an associate professor at Uppsala University, Sweden, and a senior researcher at the Swedish Defence Research Agency (FOI), Sweden. He is the author and co-author of more than 60 international journal and conference publications.


Fredrik Gustafsson is professor in Sensor Informatics at Department of Electrical Engineering, Linköping University, since 2005. He received the M.Sc. degree in electrical engineering 1988 and the Ph.D. degree in Automatic Control, 1992, both from Linköping University. He is the author of five books, more than 600 scientific papers, papers and some 30 patents. His current h-index is 71 (Google Scholar). He has supervised 25 Ph.D. and more than 200 master theses.

He was an associate editor for IEEE Transactions on Aerospace and Electronic Systems 2010–2012, and EURASIP Journal on Applied Signal Processing 2007–2012. He was awarded the Arneberg prize by the Royal Swedish Academy of Science (KVA) 2004, elected member of the Royal Academy of Engineering Sciences (IVA) 2007, and elevated to IEEE Fellow 2011. In 2014, he was awarded a Distinguished Professor grant from the Swedish Research Council. He was an adjunct entrepreneurial professor at Twente University 2012–2013.

He was awarded the Harry Rowe Minno Award 2011 for his tutorial “Particle Filter Theory and Practice with Positioning Applications”, which was published in the AESS Magazine in July 2010. This paper was ranked 4 out of 13077 articles published in the category IEEE Engineering Aerospace between 2010 and 2014 based on Web of Science citation data. He was a co-author of “Smoothed Approximate Uncertainty in Neural Network Predictions” that received the Automatica paper prize in 2014.

He is a co-founder of the companies NIRA Dynamics (automotive safety, including tire pressure monitoring systems found in about 100 million cars today), Softube (plug-ins used in tens of thousands music studios), and Senion (indoor navigation for smartphones deployed in more than 30 countries in all six continents), now part of Verizon.


