How do Bayesian Neural Networks handle input noise?

Thomas Neher
Friedrich-Alexander-University Erlangen-Nuremberg
Erlangen, Germany
thomas.neher@fau.de

***Abstract*—In this paper, the behavior of Bayesian Neural Networks (BNN) under the influence of additional input noise is examined. Critical applications in AI such as medical diagnostics and autonomous driving rely on a measure of confidence of predictions, since wrong predictions might constitute fatal problems. While recent research presents several methods for estimating the (un-)certainty of neural network predictions, the behavior of noise in the data has remained uninvestigated for BNNs: Is noise in the inputs reflected in uncertainty in the outputs? In this paper, a standard NN is compared with an implementation of BNNs for a one-dimensional regression task, introducing various levels of input noise and comparing their respective prediction uncertainty. It could be shown that BNNs' outputs reflect the uncertainty in the inputs: larger input noise corresponds to larger output uncertainty. However, runtime for training and predicting is longer for BNNs. Our results imply that BNNs are adequate for use in time-uncritical applications with need for rejection of low-confidence outputs.**

***Index Terms*—Uncertainty, Bayesian methods, Neural networks, Additive noise**

I. Introduction

In supervised learning problems, neural networks (NN) often yield high-confidence outputs even for yet unseen data [1]. However, predictions become unreliable when inputs are out of the training distribution or corrupted by noise [2]. Reliable predictions become important in critical applications such as medical diagnosing, autonomous vehicles, or high frequency trading [3]. Trusting unreliable predictions in these domains easily lead to problems with grave or even unforeseeable consequences. In order to address this problem, a framework for estimating the confidence of neural networks' outputs has been presented: Bayesian Neural Networks (BNN) [1], [4]. Within this framework, outputs come with certainty measures. However, literature has not yet examined the influence of noise in the inputs on the confidence of BNN outputs. When inputs are corrupted by noise, statements from our NN are expected that are less reliable.

Uncertainty estimation in supervised learning problems is an active area of research in deep learning. Various methods of estimating uncertainty of predictions have been presented in recent literature, including uncertainty estimations based on dropout [5] or on deep ensembles [6]. Locquerio et al. proposed a general framework for uncertainty estimation for out-of-distribution data based on belief networks and Monte-Carlo sampling [2]. DeVries and Taylor introduced a method of obtaining a measure of uncertainty without the need to consider the output distribution [7].

The largest part of literature in NN uncertainty estimation, however, is concerned with Bayesian NNs. They have first been introduced by MacKay et al. [4] and Neal [8]. In their works, they established advantages of BNN such as robustness to overfitting and the ability to learn from small data sets. While Blundell et al. presented the notion of replacing fixed weights with probability distributions in order to enable the Bayesian framework in the context of NNs in detail, Gales and Malinin proposed the introduction of a prior network over predictive distributions [9]. For an overview of the current state of the art regarding Bayesian Deep Learning, see Gal [3].

While recent literature has presented a multitude of approaches towards estimating uncertainty in general, it has not, however, evaluated the influence of additional noise in the inputs for a Bayesian Neural Network for a supervised regression task. Whereas noise in the inputs corresponds to a higher level of uncertainty in the inputs, the question remains, whether the higher degree of uncertainty is also reflected in the outputs. Thus, the research question of this paper is: Does noise in the inputs lead to higher levels of uncertainty in the outputs? More precisely, is the framework of Bayesian Neural Networks able to indicate uncertainty in the output when there is uncertainty in the data?

The remainder of the paper is structured as follows: Section 2 introduces the methods used within the scope of this paper, including a description of the data set and the theoretical foundation of this paper. Also, the course of actions taken is described. In section 3, the empirical results of an exemplary regression problem are presented and runtime of plain NN vs. Bayesian NN is compared. A discussion of these results follows in section 4, where these results are set into relationship with other findings in the literature. Section 5 contains a conclusion of the findings of this paper.

II. Methods

The data used within the scope of this paper is the UCI air quality dataset [10]. It consists of hourly averaged responses of 5 metal oxide chemical sensors and corresponding ground truth measurements. Both types of data were recorded by Chemical Multisensor devices, located on the field in an Italian city. Measurements started in March 2004 and concluded in

February 2005, resulting in one complete year of recordings. For the purposes of this paper, the data set was used for an examplary supervised regression problem.

The regression problem is defined here as a probabilistic model $P(\mathbf{y}|\mathbf{x}, \mathbf{w})$, which assigns a probability for given inputs $\mathbf{x} \in \mathbb{R}^p$ to possible outputs $\mathbf{y} \in \mathbb{R}$, given weights $\mathbf{w}$. First, a standard regression NN is created in order to later on compare it to BNN with respect to runtime. In NNs, inputs are mapped onto the outputs by passing through successive layers in which linear and non-linear transforms are performed. The weights $\mathbf{w}$ are learned using Maximum Likelihood Estimation (MLE). Given the dataset $\mathcal{D}$ of training examples $(\mathbf{x}_i, \mathbf{y}_i)_i$, weights are obtained by calculating $\mathbf{w}_{ML}$:

$$\begin{aligned}\mathbf{w}_{ML} &= \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) \\ &= \arg\max_{\mathbf{w}} \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}).\end{aligned}$$

In standard NNs, this set of weights is typically searched for using backpropagation, i.e. successive forward and backward passes. In the course of training, gradient descent updates the weights iteratively, using a learning rate η that is responsible for the update size:

$$\mathbf{w}_{new} \leftarrow \mathbf{w}_{old} - \eta \Delta_w$$

Instead of fixed weights, BNNs introduce probability distributions for the weights. The aim then is to calculate a suitable posterior distribution. Thus, within the Bayesian framework, instead of MLE, Maximum A Posteriori estimation is used.

$$\begin{aligned}\mathbf{w}_{MAP} &= \arg\max_{\mathbf{w}} \log P(\mathbf{w}|\mathcal{D}) \\ &= \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) + \log P(\mathbf{w}).\end{aligned}$$

Since obtaining this posterior analytically is infeasible, variational learning has been used as proposed by Hinton and Van Camp [11]. For this, an approximating distribution $q(\mathbf{w}|\theta)$ is assumed. The task is then to find the parameters θ so that the Kullback-Leibler (KL) divergence of q and P is minimized:

$$\mathrm{KL}[q(\mathbf{w}|\boldsymbol{\theta})||P(\mathbf{w}|\mathcal{D})] = \int q(\mathbf{w}|\boldsymbol{\theta}) \log \frac{q(\mathbf{w}|\boldsymbol{\theta})}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})} \mathrm{d}\mathbf{w}$$

Friston et al. [12] have shown that this corresponds to the minimization of a quantity termed variational free energy. It is denoted as

$$\mathcal{F}(\mathcal{D}, \boldsymbol{\theta}) = \mathrm{KL}[q(\mathbf{w}|\boldsymbol{\theta})||P(\mathbf{w})] - \mathbb{E}_{q(w|\theta)}[\log P(\mathcal{D}|\mathbf{w})]$$

Minimizing this cost function with respect to θ directly, again, is infeasible. Instead, stochastic variational inference is used as introduced by Hoffman et al. [13]. In combination with the re-parametrisation trick introduced by Opper and Archambeau [14] and Rezende [15], one can make use of Monte-Carlo-sampling to approximate the cost function. Let $\mathbf{w}_i$ be the i-th MC sample drawn from the variational posterior $q(\mathbf{w}_i|\theta)$. Then, one can write

$$\mathcal{F}(\mathcal{D}, \boldsymbol{\theta}) \approx \sum_i \log q(\mathbf{w}_i|\boldsymbol{\theta}) - \log P(\mathbf{w}_i) - \log P(\mathcal{D}|\mathbf{w}_i).$$

If one takes their variational posterior to be normally distributed, a set of weights $\mathbf{w}$ can be obtained by sampling a unit Gaussian and transforming it to mean μ and standard deviation σ. With $\boldsymbol{\theta} = (\mu, \sigma)$, our optimization procedure looks as follows:

1) Forward pass: Within the forward pass, a set of weights is sampled as described above. With this, the value of
$$f(\mathbf{w}, \boldsymbol{\theta}) = \log q(\mathbf{w}|\boldsymbol{\theta}) - \log P(\mathbf{w})P(\mathcal{D}|\mathbf{w})$$
is calculated.
2) Backward pass: Instead of updating $\mathbf{w}$ directly, we calculate gradients with respect to μ and σ and update them:
$$\begin{aligned}\Delta_\mu &= \frac{\partial f(\mathbf{w}, \boldsymbol{\theta})}{\partial \mathbf{w}} + \frac{\partial f(\mathbf{w}, \boldsymbol{\theta})}{\partial \mu}, \\ \Delta_\sigma &= \frac{\partial f(\mathbf{w}, \boldsymbol{\theta})}{\partial \mathbf{w}} + \frac{\partial f(\mathbf{w}, \boldsymbol{\theta})}{\partial \sigma}.\end{aligned}$$
With this, we can update our variational parameters to approximate our posterior distribution:
$$\begin{aligned}\mu &\leftarrow \mu - \eta \Delta_\mu, \\ \sigma &\leftarrow \sigma - \eta \Delta_\sigma.\end{aligned}$$

The base line plain NN consisted of a simple, but expedient architecture based on a fully connected (FC) layer, a ReLU activation layer and another FC layer, successively. The BNN consisted of the same architecture, i.e. the same layers as the base line NN. Our implementation of the BNN used PyTorch as deep learning framework in combination with Pyro, a probabilistic programming framework [16]. Pyro contains wrappers for PyTorch that enable handling variational inference by including methods for various kinds of Monte Carlo methods. This framework has been used for the following regression task: The input for the regression is a sensor measurement for the metal oxide NOx from the UCI air quality data set. The regression task was then to predict the ground truth from these measurements. After training of the BNN, i.e. when an approximation of the posterior distribution is obtained, one can predict outputs of unseen data $\mathbf{x}$ via

$$P(y|\mathbf{x}, \mathcal{D}) = \int P(y|\mathbf{x}, \mathbf{w}) P(\mathbf{w}|\mathcal{D}) \mathrm{d}\mathbf{w}.$$

Again, this integral is intractable, so that Monte-Carlo-estimation can be used to get an unbiased estimate by sampling from the variational posterior:

$$P(y|\mathbf{x}, \mathcal{D}) \approx \frac{1}{M} \sum_{i=1}^{M} P(y|\mathbf{x}, \mathbf{w}_i) \quad , \quad \mathbf{w}_i \sim q(\mathbf{w}|\boldsymbol{\theta}).$$

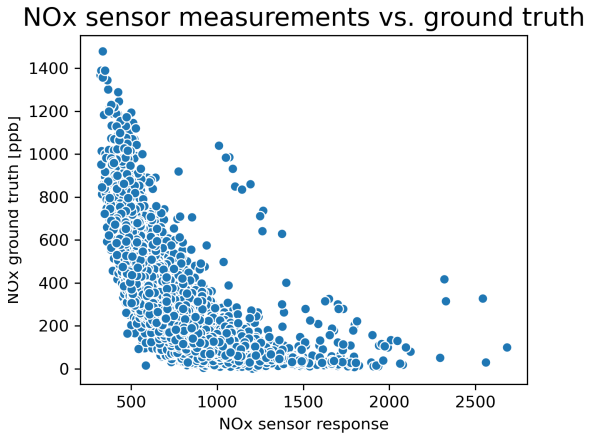

Fig. 1. Sensor measurements vs. ground truth for the metal oxide NOx. An inverse relationship can be observed. Y-axis is unitless.

By obtaining multiple predictions, statistics can be identified such as mean of predictions, standard deviation or interdecile range (IDR), i.e. the distance between 9th and 1st decile.

In order to determine the effect of noise on the predictions, different levels of Gaussian noise were added to the inputs and one network each for different amounts of noise was trained:

$$\tilde{\mathbf{x}} = \mathbf{x} + \epsilon, \quad \epsilon \sim N(0, \sigma^2),$$

where larger magnitudes of σ^2 correspond to larger amounts of noise. For each of these BNNs, 300 predictions were created and mean and IDR obtained.

III. Results

Regression was performed using one of the measured chemicals that is considered highly relevant for air pollution, namely NOx. NOx is a term for the subsumption of nitrogen oxides that are most relevant for air pollution, i.e. nitric oxide (NO) and nitrogen dioxide (NO_2). While exploratory data analysis displayed high magnitude correlations between all metal chemical sensor measurements and their respective ground truths, this correlation is negative (-0.67) for NOx. This negative correlation is indicated in the scatter plot of NOx measurements of a given point in time on the x-axis and their corresponding ground truths on the y-axis in fig. 1.

Runtime comparison showed that training of the plain NN and generation of 300 predictions lasted 5.1 seconds, on average, whereas the same process lasted 11.6 seconds for the BNN, on average. Fig 2. shows predictions of the BNN for various levels of input noise (no noise, $\sigma^2 = 0.1$, $\sigma^2 = 0.2$, respectively). While the original input has been normalized to collapse to the range $[0, 1]$, predictions were taken for values in the interval $\mathbf{x}_{pred} \in [-0.2, 1.2]$ in order to investigate behavior of the BNN for yet unseen data. For the noiseless case on the left, the maximum interdecile range (IDR) of predictions within the range of $\mathbf{x}_{pred}$ is 0.12. In the case of $\sigma^2 = 0.1$, i.e. for slightly noisy data, maximum

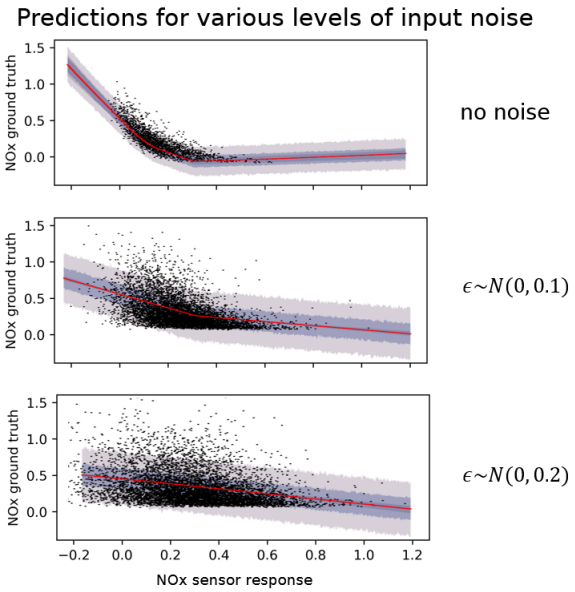

Fig. 2. Predictions of BNN with varying amounts of input noise. Axes have been normalized. Red line indicates mean of predictions, grey area depicts inter decile range of predictions, black dots are noisy training data. Top: no noise, center: $\sigma^2 = 0.1$, bottom: $\sigma^2 = 0.2$

IDR is 0.45, whereas for more amplified noise of $\sigma^2 = 0.2$, maximum IDR is 0.48.

IV. Discussion

In order to answer the question whether noise in the inputs leads to higher levels of uncertainty in the outputs, the framework of Bayesian NNs was used to set up a supervised regression problem with uncertainity estimates. For this, a metal oxide sensor measurement from the UCI air quality data set was mapped onto ground truth values of the same metal oxide.

Results showed a high, yet negative correlation between the sensor measurements of hourly averaged NOx and their corresponding ground truth values. Since both measurements make statements about the concentration of NOx in the air, this seems counterintuitive at first. However, this can be explained by the fact that while the ground truth directly measures the concentration of particles in ppb (parts per billion), the sensor does not. Instead, the sensor yields higher values in the case of absence of NOx particles in the air. Thus, high ground truth values correspond to low sensor values.

Within BNNs, fixed valued weights that occur in common NNs are replaced with probability distributions. Assuming a Gaussian prior, during training, the parameters of these weight distributions are updated in order to approximate the posterior distribution. When comparing runtime of the plain NN and the BNN approach, it could be seen that the BNN needs more than double the time for training the network and predicting 300 values than the plain NN, on average. This is mainly explained by the amount of parameters needed in each of these cases and

by the various uses of Monte-Carlo estimation. Since in BNNs, fixed valued weights are replaced with posterior distributions, the amount of parameters that are optimized in the course of the training process is doubled: Whereas in the plain NN, the weights $\mathbf{w}$ are updated directly via gradient descent, for BNNs, the set of parameters θ has to be updated. For Gaussian posteriors, this means that both μ and σ, i.e. mean and standard deviation of the approximative distribution, have to be approximated. Also, for stochastic variational inference, i.e. for approximation of the cost function as well as for the prediction integral, Monte Carlo methods are used. The use of these is more time intensive compared to standard NNs. This longer runtime implies that BNN might not be suited well for applications in which runtime is crucial. This constitutes a possible disadvantage and limit of the BNN approach.

On the other hand, replacing fixed valued weights with a posterior distribution introduces another effect to the network: Since for each evaluation of the network, weights are drawn from the posterior, such an approach corresponds to training an ensemble of networks [1]. Other ensemble methods often use multiple parameters [1]. In this respect, merely having to double the amount of parameters needed for standard NNs can be seen as an advantage of the BNN approach.

Based on the trained BNN, it was possible to predict outputs. We did so by taking 300 predictions, each time sampling from the variational posterior, and calculating characterizing statistics from these predictions, i.e. mean and interdecile ranges. Regarding the differences in interdecile ranges (IDR) for various amounts of input noise, it could be seen that uncertainity in ouputs rises with larger amounts of noise in the inputs. This is reflected in the graphical depiction of fig. 2: Whereas the maximum IDR is 0.12 for the noiseless case, it rises to 0.45 for a small amount of Gaussian noise ($\sigma^2 = 0.1$). For an additionally elevated noise of ($\sigma^2 = 0.2$), it further rises to 0.48. Graphically, this is depicted in the increase of grey area surrounding the prediction mean for larger amounts of input noise. With respect to the research question posed above, results show that BNNs do behave as expected: The uncertainity in inputs is reflected in the uncertainity in outputs.

Tthe results also showed, however, that while the grey area indicating the IDR becomes larger with increasing input noise, they remain to a high degree constant in size throughout the whole input range of one single trained BNN. The expected behavior here was that in areas outside the training examples, the IDR attains larger values than in areas of already observer data. Such behavior is also displayed in the literature [1]. This might be due to the fact that training was not stopped soon enough so that effects of overfitting occured. Indeed, within an experimental setup, stopping training early yielded results that showed the expected behavior. Subsequent work should undertake corrective measures to prevent such problems from arising.

Lastly, within the scope of this paper, only a simple regression task from one measurement onto the other was investigated. The time structure of the given data was not taken into consideration, i.e. the fact that the data does not only consist of pairs of inputs and outputs, but that they originate from time series data. Taking this sequential structure into account, there might be ways to cope with the noise in the inputs so that the uncertainty in the outputs is reduced or even vanishes completely. Applying such noise-reducing methods and investigating them with methods of uncertainty estimation might present an interesting endeavor.

V. Conclusion

The purpose of this paper was to investigate, whether noise in the input leads to higher uncertainty in outputs for a given regression problem within the framework of Bayesian NNs. Indeed, the results seem to corroborate this assumption: Higher levels of input noise corresponded to a larger degree of overall prediction uncertainty. This, in turn, implies that reduction of input noise is a valuable approach if one's aim is to raise the confidence of BNNs output, even for yet unseen data. Thus, using BNNs should be taken into consideration for tasks where noise is present, since BNNs will mirror this noise by displaying more uncertain predictions.

When runtime is a crucial factor of system design for any given task, however, usage of BNNs should be reconsidered, since the higher amount of parameters and application of Monte-Carlo estimation methods leads introduces highly elevated runtimes. In these cases, other methods that do not rely on such applications are preferable.

While the findings of this work apply to the task of one-dimensional regression, this is the only task that has been performed within the scope of this paper. Future work might also incorporate more complex regression tasks such as multiple or multivariate regression. Furthermore, since the given air quality data set consists of time series data, it might be possible to draw conclusions from including the time strucutre of this data. A possible direction of research is given by the work of [17], who set up a framework denoted as Recurrent Bayesian Neural Networks.

Lastly, while in this paper, BNNs were investigated as a special type of framework for uncertainty estimation, there is a variety of other methods that serve the same purpose. Thus, comparison of these methods is advisable for future work in this branch of research. Since different tasks have different requirements, it will be interesting to examine, which methods meet which requirements. After all, artificial intelligence will play an ever-increasing role in the years to come. With the efforts mentioned above, research will make sure that critical applications involving machine learning such as medical diagnostics and autonomous driving will remain safe and secure.

References

[1] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight Uncertainty in Neural Networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, vol. 37, 2015, pp. 1613–1622.

[2] A. Loquercio, M. Segu and D. Scaramuzza, "A General Framework for Uncertainty Estimation in Deep Learning," in *IEEE Robotics and Automation Letters*, vol. 5, no. 2, 2020, pp. 3153-3160.

[3] Y. Gal, "Uncertainty in deep learning." Ph.D. Dissertation. University of Cambridge, 2016.

[4] D. J. C. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Computation*, vol. 4, no. 3, 1992, pp. 448-472.

[5] Y. Gal, Yarin, and G. Zoubin. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." international conference on machine learning, 2016.

[6] L. Balaji, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, 2017.

[7] T. DeVries, and G. W. Taylor, "Learning confidence for out-of-distribution detection in neural networks," *arXiv preprint*, 2018.

[8] R. M. Neal, *Bayesian learning for neural networks*, New York City, NY: Springer Science and Business Media New York, 1996.

[9] A. Malinin, and M. Gales. "Predictive uncertainty estimation via prior networks," *Advances in Neural Information Processing Systems*, 2018.

[10] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia, "On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario", Sensors and Actuators B: Chemical, vol. 129, 2008, pp. 750-757.

[11] G. E. Hinton, and D. Van Camp. "Keeping the neural networks simple by minimizing the description length of the weights," *Proceedings of the sixth annual conference on Computational learning theory*, 1993, pp. 5-13.

[12] K. Friston, J. Mattout, N. Trujillo-Barreto, J. Ashburner, and W. Penny, "Variational free energy and the Laplace approximation," *Neuroimage*, vol. 34, no. 1, 2007, pp. 220-234.

[13] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *The Journal of Machine Learning Research* vol. 14, no.1, 2013, pp. 1303-1347.

[14] M. Opper, and C. Archambeau, "The variational Gaussian approximation revisited," *Neural computation*, vol. 21, no.3, 2009, pp. 786-792.

[15] D. J. Rezende, M. Shakir, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models." *arXiv preprint*, 2014.

[16] E. Bingham, et al., "Pyro: Deep universal probabilistic programming," *The Journal of Machine Learning Research* vol. 20, no. 1, 2019, pp. 973-978.

[17] M. Fortunato, C. Blundell, and O. Vinyals, "Bayesian recurrent neural networks." *arXiv preprint*, 2017.