Universiteit Leiden

Master Computer Science

Simulation study of Uncertainty Quantification on classification tasks using Empirical Bayesian Deep Neural Network

Name:	Maxime Casara
Student ID:	s2465124
Date:	08/09/2021
Specialisation:	Advanced Data Analytics
Supervisor:	Stefan Franssen (MI)
Supervisor:	Dr. Jan N. van Rijn (LIACS)
Supervisor:	Dr. Daniël M. Pelt (LIACS)

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

A simulation study of Uncertainty quantification in classification tasks using Empirical Bayesian Deep Neural Networks

Maxime Casara s2465124

8th Setpember, 2021

Contents

Abstract

This report presents a study of uncertainty quantification of deep neural networks on classification tasks. The main contribution is the introduction of an empirical Bayesian method to compute credible intervals for a deep neural network. We tested the method on a binary classification task and compared the results with an ensemble learning approach for uncertainty quantification. We showed that the Empirical Bayesian Neural Network (EBNN) performed comparatively well to a bootstrapping method using significantly less running time. An attractive feature of Empirical Bayesian Neural Network is the scalability with an increasing number of training points, which is precisely the main weakness of ensemble learning methods. The motivation to develop an Empirical Bayesian Neural Network was to take advantage of the Bayesian framework for uncertainty quantification and apply it to deep neural networks. Unlike fully Bayesian networks, only the last layer of the network is built with Bayesian weights while the other weights are regular point estimators. This significantly addresses the issue of computing the complex true posterior distribution of the whole neural network while providing correct credible intervals guaranteed by theory.

1 Introduction

Given a trained neural network estimator on a classification task, which statistical tools can be used to quantify the uncertainty of the predictions? Although neural networks have proved to be efficient for numerous predictions tasks, their behavior has yet to be fully understood. We observe that building and optimizing neural networks is usually done by exploring hypotheses rather than following theoretical motivations. This strategy of trial and error is the most practical and is justified by the lack of theoretical results on how to build optimal neural networks. Inexperienced practitioners most often use design patterns [16] to guarantee good behaviors on the approximation. However, it comes at the cost of additional computational expenses and a lack of theoretical guarantees. For example the practice of tuning hyperparameters by trying many configurations [21] is computationally expensive and doesn't provide any guarantees to converge to the global minima.

As of today, it is out of the question to ask for theory-driven results on all aspects of neural networks. However, it is quite engaging to work in these directions. This project presents a simulation study built upon theoretical guarantees of a method for uncertainty quantification in deep neural networks. Most results are based on Johannes Schmidt-Hieber's work [14] on nonparametric regression, and we explore the theoretical guarantees on classification tasks. To do so, we set up a binary classification task with controlled complexity and challenge 3 methods of calculating uncertainty quantification of neural networks. The questions we aim to answer in this study are the following:

- How do theory driven methods perform in comparison to other standard methods of uncertainty quantification?
- Do theoretical guarantees come at the cost of performance?

We start by introducing the relevant notions of uncertainty quantification, then present a thorough justification of the Bayesian framework, which our method is based on. We summarize Johannes Schmidt-Hieber's [14] results on neural network and their ability to approximate functions, and finally present the simulation study on uncertainty quantification. The experiments

were performed with the help of ALICE High-Performance Computing joint facility of Leiden University and Leiden University Medical Center.

1.1 Uncertainty quantification

Uncertainty quantification regroups the methods of evaluating the uncertainty of predictions in statistical inference problems. In a most straightforward approach, uncertainty quantification statements can be made by calculating the variance and the standard deviation error of a probability distribution. However, these parameters are not always easy to recover in practice. As more and more critical applications use deep neural networks for high-risk decisions like medical imaging [12] and image recognition for self-driving cars [3], there has been an increase of interest in uncertainty quantification in deep neural networks. Moloud Abdar et al. [1] reviewed two classical methods for uncertainty quantification: ensemble learning and Bayesian methods. More recently (July 2021) a survey [7] also reviewed these two methods as well as test-time augmentation methods and single network deterministic methods. In the context of our project, we will be interested in Bayesian and ensemble learning techniques.

In Bayesian methods, we evaluate the credible intervals of the predictions of the deep neural network. Credible intervals are statements on the density of the posterior distribution. For example, a 95% credible interval on a parameter's distribution is an interval that contains 95% of the density of the distribution. The interpretations of these objects are not always evident, because for example, they depend on the choice of prior.

Without characterizing uncertainty, we would trust the neural network to have found a coherent representation to approximate the true function. While the prediction can be accurate, we have to assess how likely we can trust the network to produce predictions when we introduce variations in the sample space. An assumption when using machine learning methods is that the sampling space is not too far from the true sample space. We generally assume that we sampled the data from the true underlying generating process, and therefore, any estimator trained on the data with satisfying accuracy would carry over well on newly generated data. In the words of [11], these learned hypotheses derived from the data give an optimistic estimate of hypothesis accuracy over future examples. It is referred to as bias in the estimate and is not countered by using validation and testing sets.

1.2 Bayesian paradigm

It is quite pragmatic to use Bayesian methods to study uncertainty quantification in that they provide built-in uncertainty quantification statements. However, they do so in the form of credible intervals which are not always fit for interpretation. In uncertainty quantification, we are interested in more regular statements such as confidence intervals, which are provided by frequentist methods and not Bayesian methods. It is of interest to evaluate how credible intervals compare to confidence intervals. To do so, we introduce in the following sections the notions at stake.

1.2.1 Bayes Theorem

The general idea behind Bayesian statistics is to give estimates in form of distributions. To illustrate this, let $X_0, ..., X_n$ be observations from a distribution of parameter θ_0. We refer to $X_0, ..., X_n$

as the realizations of a random variable X that follows a distribution of parameter θ_0. Now let's consider θ_0 to be itself the realization of a random variable θ from a distribution Π_0. We could in principle continue the same treatment and define Π_0 as the realization of another random variable Π, but let's first examine the role of the distribution Π_0 over the observations $X_0, ..., X_n$. By the law of total probability and under some assumptions, we can express the marginal distribution of X as the sum over all $\theta_0, ..., \theta_n$ of the product of the probability of X given $\theta_0, ..., \theta_n$ with the probability of realization of $\theta_0, ..., \theta_n$. Let's derive this result from topology as in [8]:

Let $(\Theta, \mathcal{B})$ be a measurable space and Π a probability measure on $(\Theta, \mathcal{B})$. Let P_θ be a regular conditional distribution on the sample space $(\mathfrak{X}, \mathcal{X})$ of the data i.e.:

- the map $A \mapsto P_\theta(A)$ is a probability measure for every $\theta \in \Theta$,
- the map $\theta \mapsto P_\theta(A)$ is measurable for every $A \in \mathcal{X}$.

Then, the pair (X, θ) has a well-defined joint distribution on the product space $(\mathfrak{X} * \Theta, \mathcal{X} \otimes \mathcal{B})$:

$$P(X \in A, \theta \in B) = \int_B P_\theta(A) d\Pi(\theta).$$

The marginal distribution of X is then defined by:

$$P(X \in A) = \int P_\theta(A) d\Pi(\theta), \quad A \in \mathcal{X},$$

where $\Pi(\theta)$ is referred to as the *prior distribution*. The *posterior distribution* is the conditional distribution:

$$\Pi(B|X) = P(\theta \in B|X), \quad B \in \mathcal{B},$$

which can be expressed by Bayes formula as:

$$\Pi(B|X) = \frac{\int_B p_\theta(X) d\Pi(\theta)}{\int p_\theta(X) d\Pi(\theta)} \quad \text{(Bayes Formula)}.$$

The posterior distribution can be seen as the probability for each parameter θ to have generated the observations X. Choosing the parameter θ with the highest probability as the hypothesis is referred as the Maximum a posteriori (MAP) estimation:

$$\hat{\theta}_{MAP}(x) = \underset{\theta}{\operatorname{argmax}} \, \Pi(\theta|X).$$

1.2.2 Bayesian and frequentist inference

There are different ways to approach a statistical problem. One option is to decide whether the analyst can make subjective statements to address the problem. However, subjective methods in science would not be considered the most formal approaches. In statistical inference, the traditional approach and in a sense formal is referred to as the frequentist approach. It is practical to

draw conclusions solely on what is observed so that another person would reach the same result given the same data. Yet, in the school of Bayesian statistics, its partisans advocate for the use of subjective statements in the form of a prior distribution. It comes naturally that any evidence as observed by humans, be it any statistical parameter on a controlled population, already is dependent on arbitrary choices and methods of measurements. Bayesian statistics merely extend this notion further and allow the observer to make assumptions on how the data is generated. Freedom to challenge the origin of the data with initial beliefs and to accept the one that is considered for some as the most reasonable approach when facing statistical problems. To give justification for Bayesian methods, the subjective statements are sometimes said to be coming from expertise and knowledge about the problem.

Let's illustrate both approaches on the coin toss example and compare the resulting confidence interval and credible interval. Suppose we have a cheated coin that has probability p to produce head and $1-p$ to produce tail. The goal is to approximate the parameter p from the observation of n tosses. Each toss is the realization of a random variable X_i which takes value 1 for head and 0 for tail. An observation is then referred as the realization of $(X_0, ..., X_n)$. The frequentist approach, as the name suggests, will consider the value of p to be the frequency of heads, and subsequently $1-p$ to be the frequency of tails. The uncertainty about the value is determined by the confidence interval:

$$I_c = \Big[p - z^* \frac{s}{\sqrt{n}}; p + z^* \frac{s}{\sqrt{n}}\Big],$$

where s is the observed standard deviation and z^* the z-score of the confidence level. As the number of observations grows, the width of the confidence interval diminishes. An observation and a model are all that is needed for a frequentist to make uncertainty statements in our example. The Bayesian approach is conceptually different: we start by saying that for $p \in]0, 1[$, an observation of n tosses could be generated by any value of p. Then we calculate how likely each value is of generating the observation, which will represent the posterior distribution of the parameter p. In addition, the Bayesian model introduces a prior belief in the form of a prior distribution. It represents how likely we believe the value of the parameter p to be. For example, if our cheated coin comes from a batch of coins we know to be of parameter $p = 0.8$, we can choose a prior distribution that puts more weight around 0.8 before any observation has been made, so that our hypothesis is initially skewed towards that value. In practice, one can omit prior belief by choosing an uninformative prior such as the uniform distribution. It is interesting to note that in this setup, a MAP estimator becomes a Maximum Likelihood (ML) estimator. In our example, the model of the likelihood is a binomial distribution of the form:

$$\mathcal{L}(X|p) = \binom{n}{h} p^h (1-p)^{n-h}.$$

We choose the prior as a beta distribution $\Pi \sim \text{Beta}(\alpha, \beta)$:

$$\text{Beta}(x, \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)},$$

where B is the Beta function:

$$B(x, y) = \int_0^1 t^{x-1}(1-t)^{y-1}dt.$$

We can now calculate the posterior distribution. The choice of this particular prior here allows us to have an analytically defined posterior. We can prove that the beta distribution is a Bayesian conjugate prior of the binomial likelihood. To do so, we prove that the posterior distribution follows a beta distribution. By Bayes formula, we write:

$$\Pi(p|X) = \frac{\mathcal{L}(X|p)\Pi(p)}{\int \mathcal{L}(X|y)\Pi(y)dy}.$$

Which in our example can be written as:

$$\Pi(p|X) = \frac{\binom{n}{h}p^h(1-p)^{n-h}p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)\int_0^1 \left(\frac{\binom{n}{h}y^h(1-y)^{n-h}y^{\alpha-1}(1-y)^{\beta-1}}{B(\alpha,\beta)}\right)dy},$$

and reduced to:

$$\Pi(p|X) = \frac{p^{h+\alpha-1}(1-p)^{n-h+\beta-1}}{\int_0^1 y^{h+\alpha-1}(1-y)^{n-h+\beta-1}dy}.$$

The denominator is the Beta function of parameter $(h+\alpha, n-h+\beta)$. Thus we have:

$$\Pi(p|X) = \frac{p^{h+\alpha-1}(1-p)^{n-h+\beta-1}}{B(h+\alpha, n-h+\beta)}.$$

This concludes:

$$\Pi(p|X) = \text{Beta}(p, h+\alpha, n-h+\beta).$$

This result allows us to plot the posterior distribution of the experiment with an algebraic form. Figure 1 shows the plot of the posterior distribution for an observation of 100 tosses and 20 heads, and with a prior distribution Beta$(1,1)$ (uniform distribution).

The cumulative distribution function (CDF) of the posterior is the regularized incomplete beta function I_x:

$$I_x = \frac{B(x;a,b)}{B(a,b)},$$

where $B(x;a,b)$ is the incomplete beta function:

$$B(x;a,b) = \int_0^x t^{a-1}(1-t)^{b-1}dt.$$

We can compute a 95% credible interval by finding the values p for which the CDF is below 2.5% and above 97.5%. Figure 2 shows the width of the credible interval $[p_{2.5\%}, p_{97.5\%}]$. In our example,

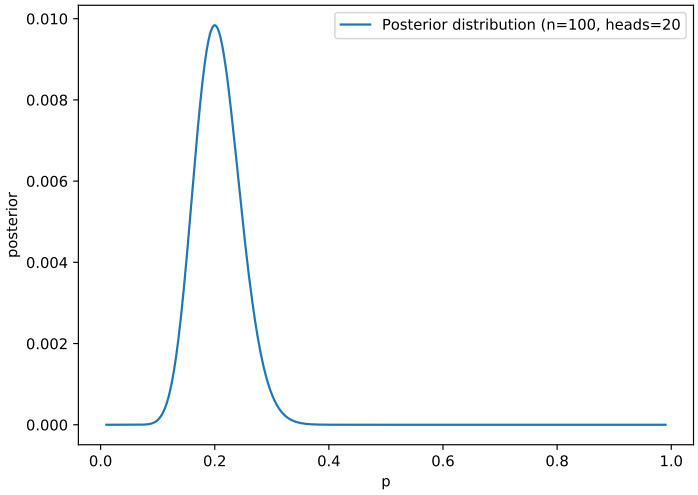


Figure 1: Posterior distribution of the coin toss example of parameter p for n=100 tosses and h=20 heads. The distribution is approximately centered at $p = 0.2$.

we find $[p_{2.5\%}, p_{97.5\%}]$to be equal to $[0.1336, 0.2896]$. For comparison we can calculate the 95% confidence interval with the frequentist approach:

$$p_{2.5\%} = \hat{p} - z^*_{95}\frac{\sqrt{\hat{p}(1-\hat{p})}}{\sqrt{n}},$$

$$p_{97.5\%} = \hat{p} + z^*_{95}\frac{\sqrt{\hat{p}(1-\hat{p})}}{\sqrt{n}},$$

where $\hat{p} = 20/100$, $z^*_{95} = 1.96$ and $n = 100$. We find $I_c = [0.1216, 0.2784]$.
We can increase the number of tosses to analyze the convergence of both approaches. Figure 3 shows the bounds of the confidence and credible intervals as the number of tosses increases. The credible interval here follows the same behavior as the confidence interval, which could give justification to interpret this object from a frequentist point of view. Note that the confidence interval is generally ill-defined for a low number of tosses.

We showed in this simple example the motivation of using the Bayesian approach to compute a credible interval. It returns a very close approximation of the confidence interval calculated with a frequentist method.

1.2.3 Parametric and nonparametric inference

In parametric inference, pre-existing models are used to bound the hypothesis space of the data generating process. In this setup, the parameters have a fixed finite number components. When

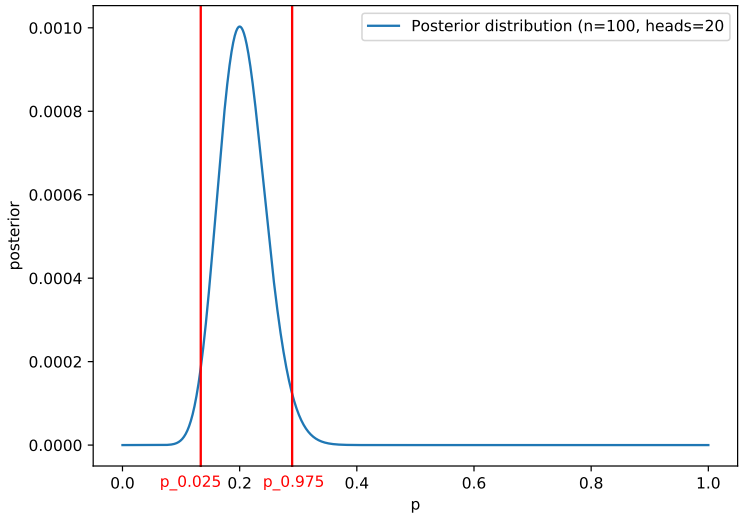


Figure 2: Width of the 95% credible interval. $p_{2.5\%}$ is the value for which I_x is below 2.5% and $p_{97.5\%}$ is the value for which I_x is below 97.5%. Note that at $p = 0.2$ the plot is not symmetrical due to the choice of prior and p being closer to the lower bound 0.

the number of components is extended to infinity, we enter the framework of nonparametric inference, which could be seen as more flexible and adaptive frameworks. One illustrative example of an application of nonparametric inference is neural networks, also known as universal function approximators [10].

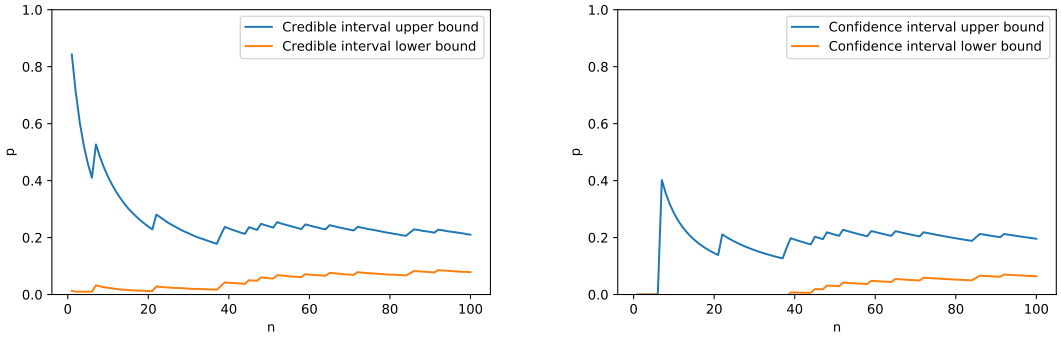
1.2.4 Bernstein-von Mises theorem

The Bernstein-von Mises theorem gives a frequentist justification of Bayesian credible intervals. Under some assumptions and in a parametric setup, the Bernstein-von Mises theorem states that a posterior distribution derived from Bayesian methods will converge to a Gaussian distribution centered around the true parameter of interest and with proper variance, given enough data. It furthermore implies that posterior distributions can provide asymptotically correct uncertainty statements with respect to the frequentist point of view.

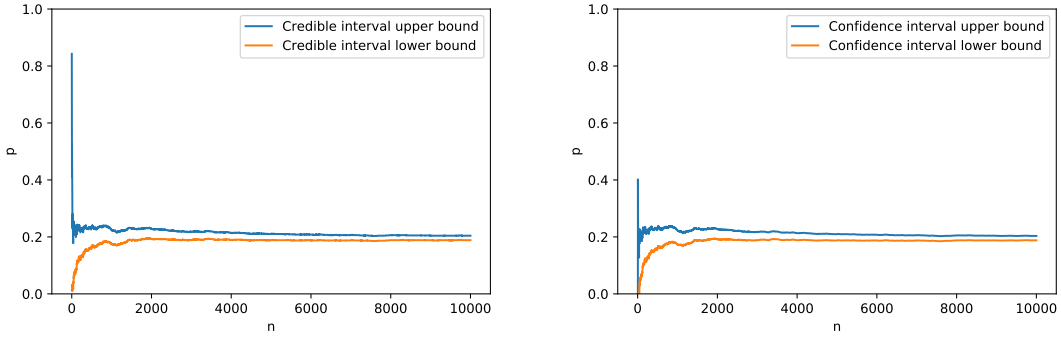
1.2.5 Posterior consistency and Contraction rate

We qualify a posterior distribution that converges to the true parameter as data grows as *consistent at* this parameter. It is a useful guarantee that our model will provide a sensible estimate and uncertainty statements as we collect more and more data. We give here the definition of posterior consistency from [8]:

Definition 1.1 (Posterior consistency) The posterior distribution $\Pi_n(\cdot|X^{(n)})$ is said to be *(weakly)*



(a) $p = 0.2$ and 100 tosses. Bayesian approximation (left), frequentist approximation (right)



(b) $p = 0.2$ and 10000 tosses. Bayesian approximation (left), frequentist approximation (right)

Figure 3: Upper bound and lower bound of the 95% frequentist confidence interval and 95% credible Bayesian interval as n grows.

consistent at $\theta_0 \in \Theta$ if $\Pi_n(U^c|X^{(n)}) \to 0$ in $P_{\theta_0}^{(n)}$-probability, as $n \to \infty$, for every neighborhood U of θ_0. The posterior distribution is said to be *strongly consistent* at $\theta_0 \in \Theta$ if this convergence is in the almost-sure sense.

Schwartz [15] theorem gives conditions that guarantee posterior consistency for the distributions of i.i.d. random variables. Notably, it is required for the support of the prior to have a non-zero density around the true parameter we need to estimate. We state here Schwartz theorem as formulated by [4]:

Theorem (Schwartz) Let Π be a prior on a metric space $\mathcal{P}$, $p_0 \in \mathcal{P}$ the true density and U the neighborhood of p_0. If:

Condition 1 (prior support): p_0 is in the KL support of prior Π i.e., for all ϵ,

$$\Pi(\{p : K(p_0, p) \leqslant \epsilon\}) > 0,$$

Where $K(p_0, p) := \int p_0 \log(\frac{p_0}{p}) d\mu$ is the Kullback-Leibler divergence between p_0 and p.

Condition 2 (testing): There exists a uniformly consistent sequence of tests for testing $H_0 : p = p_0$ vs $H_1 : p \in U^c$: that is, there exist ϕ_n such that:

$$P_0^{(n)}(\phi_n) := \int \phi_n dP_0^{(n)} \xrightarrow{n\to\infty} 0,$$

$$\sup_{p\in U^c} P^{(n)}(1-\phi_n) = \sup_{p\in U^c} \int (1-\phi_n) dP^{(n)} \xrightarrow{n\to\infty} 0,$$

where $\phi_n : \mathbb{X}^n \to [0,1]$ is a test function.
Then the posterior is consistent at p_0, i.e., for every neighborhood U of p_0,

$$\Pi(U|X_1, ..., X_n) \xrightarrow{n\to\infty} 1, \quad P_0^{(\infty)} - a.s.$$

This theorem has been subject to many extensions allowing broader applicability, such as for non-identically distributed random variables in nonparametric regression setup [5]. While an important guarantee, posterior consistency is not always the most sought after by Bayesian methods. We are also interested in the speed at which our estimator approaches the neighborhood of the true parameter of the distribution of the data, referred to as the contraction rate. We will refer to the contraction rate per its definition given by [8]. Let $\Pi_n(\cdot|X^{(n)})$ be versions of the posterior distributions relative a prior Π_n.

Definition 1.2 (Contraction rate) A sequence ϵ_n is a *posterior contraction rate* at the parameter θ_0 with respect to the semimetric d if $\Pi_n(\theta : d(\theta, \theta_0) \geq M_n \epsilon_n | X^{(n)}) \to 0$ in $P_{\theta_0}^{(n)}$-probability, for every $M_n \to \infty$. If all experiments share the same probability space and the convergence to zero takes place almost surely $[P_{\theta_0}^{(\infty)}]$, then ϵ_n is said to be a *posterior contraction rate in the strong sense*.

The notion of contraction rate in the Bayesian framework is similar to the notion of the convergence rate of frequentist methods. In deep neural networks, it corresponds to the speed at which the predictions estimate the true function as the number of data points increases. It has theoretical interest and has been studied for example by Sanjeev Arora et. al [2] for linear activation functions. In the paper, they proved the convergence of gradient descent to global minima at a linear rate, only under some assumptions on initialization. Nicholas Polson and Veronika Rockova [13] showed minimax contraction rate of deep Bayesian ReLU networks for $\alpha -$ Hölder functions in regression setup.

To summarize, given training examples, a learner and a true parameter θ_0 we want to estimate, we are now interested in the consistency of the learner's posterior distribution at θ_0 and the posterior contraction rate at θ_0. We remind that having a contraction rate without consistency has no purpose. Therefore, we imply consistency when making statements on a Bayesian learner's contraction rate. Results on these properties will be presented in Section 2.1.

1.2.6 Priors

A prior distribution represents our beliefs on a given statistical problem, in terms of a distribution of the parameters of the data. The choice of prior can influence the behavior of the model, notably

the posterior consistency and the contraction rate. It also bounds the hypothesis space. As seen in Schwartz's theorem, for a learner's posterior distribution to be consistent with the true hypothesis, it is required that the prior has at least some density around the neighborhoods of the true parameter value. Usually, in Bayesian frameworks, we try to choose a prior that is a conjugate of the likelihood distribution, like in the coin toss example in Section 1.2.2. This simplifies the computation of the posterior, which will inherit the algebraic form of the prior and likelihood. Unfortunately in the classification setup, no conjugate prior is readily available.

1.3 Related Work & Motivation

The authors of [19] inspect the peculiar misclassifications of deep neural networks for image classification. They introduce the confusion error, which happens when two classes are close to each other, and the bias error which is the regular classification errors of a network's wrong inferences. In practice, uncertainty quantification allows us to assess the quality of a network's inferences, and therefore the quality of a network's predictions.

Moloud Abdar et al. [1] review two of the most conventional techniques of calculating uncertainty quantification: Bayesian approximation and ensemble learning. Their study also includes a wide overview of the current (June 2020) uncertainty quantification motivations, applications, and scientific discussions. The main challenges listed were "the absence of theory, absence of causal models, sensitivity to imperfect data and computational expense".

Nicolas Ståhl et al. [17] study the uncertainty quantification in neural networks when the inputs diverge greatly from the training data. They argue that Artificial Intelligence programs in real-world applications must be consistent even in critical settings, and a way to measure this consistency is to calculate its uncertainty on predictions. They want to show that the errors on prediction are high in uncertainty. While the Bayesian network proved to be the most efficient, it did not capture any difference in uncertainty between misclassified data and irrelevant data. Thus, they deemed the method insufficient.

Lior Hirschfeld et al. [9] show that none of their methods to calculate uncertainty quantification for regression task on molecular property prediction performed sufficiently well. In line with [17] and [1] they argue that there is exist no baseline model to calculate uncertainty quantification accurately, but only task-specific methods.

In [20], Matias Valdenegro-Toro advocates the use of Bayesian networks for computer vision applications because of their good approximations of uncertainty quantification.

All in all, most related works worry about one thing: the untrustworthiness of neural networks. This constitutes a problem as more and more critical applications rely on deep neural networks. It comes obvious that it is difficult to trust a program operating in a *black box*. However, when a solution is a successful predictor, we rather praise its results than question its methods. In this project, we carry out a simulation study based on theory-driven results from [6] and [14], which motivate the creation of the Empirical Bayesian Network to perform uncertainty quantification statements.

2 Model

2.1 Neural network

In this section, we consider neural networks as mathematical functions. We summarize Johannes Schmidt-Hieber's results [14] on nonparametric regression using deep neural networks, which shows how good neural networks are at approximating functions from a theoretical point of view.

2.1.1 Definitions and hyperparameters

As in [14], given a nonparametric regression model of the form:

$$Y_i = f_0(X_i) + \epsilon_i, \quad i = 1, ..., n.$$

Where $X_i \in [0,1]^d$ are independent and identically distributed random variables (i.i.d.), ϵ_i are i.i.d. noise variables following a normal distribution with mean zero and variance one, and $Y_i \in \mathbb{R}$ are the responses of the model. The goal is to describe the characteristics of a deep neural network estimator that recovers the function f_0 using the data (X_i, Y_i).

To give a mathematical definition of a neural network, we must formalize its architecture. We define neural network of architecture (L, p) the networks which have L number of hidden layer(s) and a width vector p of size $L+2$, where $p_0, ..., p_{L+1}$ are the width of the layers $L_0, ..., L+1$.

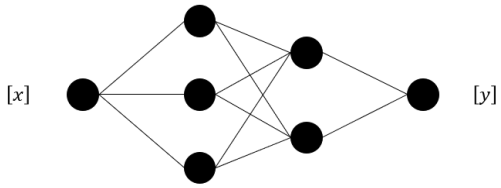


Figure 4: (L=2,p=(1,3,2,1)) Neural Network

The mathematical definition of a neural network with v-shifted ReLU activation functions is given by:

$$f : \mathbb{R}^{p_0} \to \mathbb{R}^{p_{L+1}}, \quad x \mapsto f(x) = W_L \sigma_{v_L} W_{L-1} \sigma_{v_{L-1}}, ..., W_1 \sigma_{v_1} W_0 x.$$

where W_i is the weight vector that connect the layer i to the layer $i+1$ and σ_{v_i} the activation function of the layer i. Given a network structure (L, p), we define the network sparsity class as:

$$\mathcal{F}(L, p, s, F) := \left\{ f \in \mathcal{F}(L, p) : \sum_{j=0}^{L} \|W_j\|_0 + \|v_j\|_0 \leqslant s, \| |f|_\infty \|_\infty \leqslant F \right\}.$$

We recall the results by Johannes Schmidt-Hieber which show how these neural networks can approximate smooth enough functions. The proof of each following lemma can be found the Appendix A and B of [14]. First, given an input x and y, we can construct a neural network to approximate xy. The evaluation of the approximation is given by the following result.

Lemma 1. For any positive integer m, there exist a network $\text{Mult}_m \in \mathcal{F}(m+4, (2, 6, 6, ..., 6, 1))$, such that $\text{Mult}_m(x, y) \in [0, 1]$,

$$|\text{Mult}_m(x, y) - xy| \leqslant 2^{-m}, \quad \text{for all } x, y \in [0, 1],$$

and $\text{Mult}_m(0, y) = \text{Mult}_m(x, 0) = 0$.

The next lemma extends the result on the product of r integers.

Lemma 2. For any positive integer m, there exist a network $\text{Mult}_m^r \in \mathcal{F}((m+5)\lceil \log_2 r \rceil, (r, 6r, 6r, ..., 6r, 1))$, such that $\text{Mult}_m^r \in [0, 1]$ and

$$|\text{Mult}_m^r - \prod_{i=1}^{r} x_i| \leqslant r^2 2^{-m}, \quad \text{for all } \mathbf{x} = (x_1, ..., x_r) \in [0, 1]^r.$$

$\text{Mult}_m^r = 0$ if one or more of the components of $\mathbf{x}$ is 0.

Then, we show that there exist neural networks that can approximate all monomials $\mathbf{x}^{\boldsymbol{\alpha}} := x_1^{\alpha_1} * ... * x_r^{\alpha_r}$. We note $|\boldsymbol{\alpha}| := \sum_l |\alpha_l|$ the degree of the monomial and $C_{r,\gamma}$ the number of monomial with degree $|\boldsymbol{\alpha}| < \gamma$. We state here the result on the approximation of monomials by neural network.

Lemma 3. For $\gamma > 0$ and any positive integer m, there exists a network $\text{Mon}_{m,\gamma}^r \in \mathcal{F}(1+(m+5)\lceil \log_2 \gamma \vee 1 \rceil, (r, 6\lceil\gamma\rceil C_{r,\gamma}, ..., 6\lceil\gamma\rceil C_{r,\gamma}, C_{r,\gamma}))$, such that $\text{Mon}_{m,\gamma}^r \in [0, 1]^{C_{r,\gamma}}$ and

$$\left|\text{Mon}_{m,\gamma}^r(\mathbf{x}) - (\mathbf{x}^{\boldsymbol{\alpha}})_{|\boldsymbol{\alpha}|<\gamma}\right|_\infty \leqslant \gamma^2 2^{-m}, \quad \text{for all } \mathbf{x} \in [0, 1]^r.$$

Finally, we show that there exist neural networks that approximate the Taylor expansion of a function.

Lemma 4. For any positive integers M, m, there exists a network $\text{Hat}^r \in \mathcal{F}\Big(2 + (m+5)\lceil \log_2 r \rceil, (r, 6r(M+1)^r, ..., 6r(M+1)^r, (M+1)^r), s, 1\Big)$ with $s \leqslant 49 r^2 (M+1)^r (1 + (m+5)\lceil \log_2 r \rceil)$, such that $\text{Hat}^r \in [0, 1]^{(M+1)^r}$ and for any $\mathbf{x} = (x_1, ..., x_r) \in [0, 1]^r$,

$$\left|\text{Hat}^r(x) - \Big(\prod_{j=1}^{r} (1/M - |x_j - x_j^l|)_+\Big)_{\mathbf{x}_l \in \mathbf{D}(M)}\right|_\infty \leqslant r^2 2^{-m},$$

where $\mathbf{D}(M) := \{x_l = (l_j/M)_{j=1,..,r} : l = (l_1, ..., l_r) \in \{0, 1, ..., M\}^r\}$ is a set of grid points. For any $x_l \in \mathbf{D}(M)$, the support of the function $\mathbf{x} \mapsto (\text{Hat}^r(\mathbf{x}))_{x_l}$ is moreover contained in the support of the function $\mathbf{x} \mapsto \prod_{j=1}^r (1/M - |x_j - x_j^l|)_+$.

These results show how deep neural networks can approximate smooth enough functions. They give theoretical guarantees on the neural networks accuracy, depending on the network architecture.

2.1.2 Convergence rate

Next, we are interested in how accurate are empirical measures at approximating the true underlying function and their convergence rate. To get the main results of the paper, we present from [14] the relative empirical error measure with respect to the network sparsity class $\mathcal{F}(L, p, s, F)$, regression function f_0 and estimator $\hat{f}$:

$$\Delta_n(\hat{f}_n, f_0) := E_{f_0}\left[\frac{1}{n}\sum_{i=1}^n (Y_i - \hat{f}_n(X_i))^2 - \inf_{f \in \mathcal{F}(L,p,s)} \frac{1}{n}\sum_{i=1}^n (Y_i - f(X_i))^2\right].$$

The relative empirical error measure is the expected difference in performance between an estimator and the best possible empirical estimator from the network sparsity class $\mathcal{F}$. We then define the prediction error bound with respect to the regression function f_0 and estimator $\hat{f}$ as in [14]:

$$R(\hat{f}_n, f_0) := E_{f_0}\left[(\hat{f}_n(X) - f_0(X))^2\right].$$

Unlike the relative empirical error measure, the prediction error bound takes into consideration the true regression function and measures the expected squared difference between an estimator and the underlying true function. The idea is to find a relationship between the relative empirical measure and the prediction error bound. One condition on the regression function f_0 is to assume that f_0 is a composition of functions g_i of Hölder smoothness β_i and belong to the function space:

$$\mathcal{G}(q, d, t, \beta, K) := \Big\{ f = g_q \circ ... \circ g_0 : g_i = (g_{ij})_j : [a_i, b_i]^{d_i} \to [a_{i+1}, b_{i+1}]^{d_{i+1}},$$
$$g_{ij} \in \mathcal{C}_{t_i}^{\beta_i}([a_i, b_i]^{t_i}, K), \quad \text{for some} \quad |a_i|, |b_i| \leqslant K \Big\},$$

where q is the number of functions, d a vector of dimension, β the smoothness, K an interval dominator and where each g_i is a t_i-variate function.

The main result of the paper states that, under some assumptions, if our estimator approaches the best empirical estimator at some rate, the prediction error also approaches at some rate the true underlying regression function. Given the effective smoothness indices:

$$\beta_i := \beta_i \prod_{l=i+1}^{q} (\beta_l \wedge 1)$$

and the rate:

$$\phi_n := \max_{i=0,\dots,q} n^{-\frac{2\beta_i}{2\beta_i+t_i}}.$$

Theorem (Johannes Schmidt-Hieber [14]) Consider the d-variate nonparametric regression model for composite regression function in the class $\mathcal{G}(q,d,t,\beta,K)$. Let $\hat{f}_n$ be an estimator taking values in the network class $\mathcal{F}(L,(p_i)_{i=0,\dots,L+1},s,F)$ satisfying:

(i) $F \geqslant max(K,1)$,

(ii) $\sum_{i=0}^{q} \log_2(4t_i \vee 4\beta_i)\log_2 n \leqslant L \lessapprox n\phi_n$,

(iii) $n\phi_n \lessapprox min_{i=1,\dots,L} p_i$,

(iv) $s \asymp n\phi_n \log n$.

There exist constants C,C' only depending on q,d,t,β,F, such that if $\Delta_n(\hat{f}_n,f_0) \leqslant C\phi_n L\log^2 n$, then

$$R(\hat{f}_n,f_0) \leqslant C'\phi_n L\log^2 n$$

In other words, *a network estimator is minimax rate optimal up to log factors if and only if the method almost minimizes the empirical risk.* Stefan Franssen's Empirical Bayesian Neural Network [6], which will be presented further in Section 3.2.2, is based on these results.

The choice to refer to the minimax is purely subjective, but appears in literature as the most rational estimator.

Definition 1.3 (minimax) An estimator $\delta^M : \mathcal{X} \to \Theta$ is called minimax with respect to a risk function $R(\theta,\delta)$ if it achieves the smallest maximum risk among all estimators:

$$\sup_{\theta\in\Theta} R(\theta,\delta^M) = \inf_{\delta} \sup_{\theta\in\Theta} R(\theta,\delta).$$

It is the estimator that minimizes the loss in the worst-case scenario.

2.2 Binary classification task

This technique allows us to create classification tasks with arbitrary complexity. Given a continuous function $f:\mathbb{R}\to[0,1]$, we draw uniformly at random n points $x_1,x_2,...,x_n$ and create for each a random variable X_i defined as:

$$X_i = \begin{cases} 1 & \text{with probability } p_i = f(x_i), \\ 0 & \text{with probability } 1-p_i. \end{cases}$$

Each random variable X_i follows a Bernoulli distribution of parameter $f(x_i)$. The goal is to first estimate the underlying f function given the observations X_i. To do so, we train a fully connected neural network with ReLU activation function on each couple (x_i,X_i). The output of the network

will be an estimate of $(p_i, 1 - p_i)$ for a given data point x_i.

The purpose of using a generated task and not an existing dataset is to have the information on the true function f. In a practical setup, we have no information on the true generating function, and therefore could only arbitrarily evaluate the accuracy of the methods. With the binary classification task, we can measure the distance between the predictions and the true function, which will be useful for our evaluation of uncertainty quantification.

3 Methods

We are going to study 3 different models to compute uncertainty quantification statements. The first two come from Bayesian methods and the third one stems from ensemble learning, which will be considered as baseline.

3.1 Ensemble learning

Bootstrap methods are simple solutions to implement that take advantage of the stochastic properties of learning algorithms. The idea is to approximate a model through repeated independent learning, each time on a resample of the original sample space and then aggregating the results. It is an attractive method for uncertainty quantification as it repetitively explores the sample space thus gives an approximation of the variance of the estimator. The weakness of the methods is their scalability as data grows. Even though the tasks can be performed in parallel and take advantage of distributed computing, their computational expenses still remain the main challenge.

3.2 Bayesian methods

3.2.1 Variational Bayesian inference

Variational Bayes methods aim to approximate the posterior distribution with an analytical expression. It alleviates the expensive task of computing the true posterior distribution. However, approximating the posterior distribution of a deep neural network remains too expensive, therefore we choose here to use variational inference on only the last layer of the neural network. This method will be referred to further as ADVI, which stands for Automatic Differentiation Variational Inference from the Pymc3 library.

3.2.2 Empirical Bayesian Neural Network

We introduce here the Empirical Bayesian Neural Network (EBNN) from [6] and their guarantees on credible interval coverage. Let $C_\alpha = B(c_\alpha, R_\alpha)$ be the credible ball of at least α a-posteriori probability with centering point c_α and radius R_α. The blown-up credible ball by a factor L is the ball $B(c_\alpha, LR_\alpha)$. Given the regression model introduced in Section 2.1.1:

Theorem (Empirical Bayesian Neural Network [6])

Let $\beta, M > 0$. Let Π be a prior. Let $\mathcal{D}_n = ((X_1, Y_1), \ldots, (X_n, Y_n))$ and $\epsilon_n = n^{\frac{-\alpha}{2\alpha+d}}$. Under some regularity assumptions:

- The posterior contracts at near minimax rate:

$$\limsup_{n\to\infty} \sup_{f_0 \in W_M^\alpha} \mathbb{E}_{f_0} \left(\Pi \left(f : \|f - f_0\|_2 \geq M_n \epsilon \log(n)^3 \middle| \mathcal{D}_n \right) \right) = 0,$$

  for all $M_n \to \infty$.

- The credible balls have uniform near optimal coverage: There exists $L_{\epsilon,\alpha}$ such that if $B(c_\alpha, R_\alpha)$ is an α-credible ball, the ball $B = B(c_\alpha, L_{\epsilon,\alpha} \log(n)^3 R_\alpha)$ satisfies

$$\liminf_n \inf_{f_0 \in W_M^\beta([0,1]^d)} \mathbb{P}_{f_0}^{(n)}(f_0 \in B) \geq 1 - \epsilon$$

- The credible balls have uniform near optimal size:

$$\liminf_n \inf_{f_0 \in W_M^\beta([0,1]^d)} \mathbb{P}_{f_0}^{(n)}(R_\alpha \leq C\epsilon_n) \geq 1 - \epsilon$$

  for some large enough $C > 0$.

The principle of the technique is to build Bayesian weights on the last layer of a fully connected neural network. To do so, we train a neural network and remove its last layer and its weights. We then draw new data points from D_n and feed them to the network. For each data point, we use their point estimates of the last hidden layer and perform Bayesian linear regression to link them to their target values. The Bayesian linear regression calculates Bayesian weights and their posterior distribution. Contrary to regular linear regression where the calculated parameters are point estimates, Bayesian linear regression assign a prior to each parameter and then optimizes their posterior distribution. By drawing points from these distributions, we effectively assemble a neural network with new weights on the last layer. The results from Stefan Franssen [6] guarantee the contraction rate of the posterior distributions to be at near minimax rate and to have uniform optimal coverage.

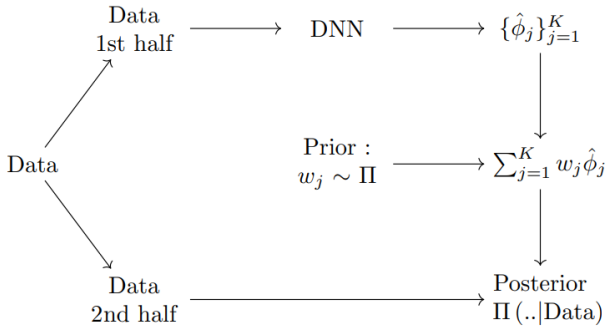


Figure 5: Description of the Empirical Bayesian Neural Network from [6]. $\hat{\phi}_1, ..., \hat{\phi}_K$ are the outputs of the last hidden layer of the least square estimator.

3.2.3 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is a sampling technique that draws data points from a posterior distribution. It is a useful method when the form of the posterior cannot be analytically defined and is expensive to compute, for example in deep neural networks. This is why we choose the Empirical Bayesian Neural Network to only compute the posterior distribution on the last layer of the network. Then we draw from the distributions with an MCMC sampler. One example of an MCMC sampler is the Metropolis-Hastings algorithm, which uses random walk with acceptance threshold to generate draws from a given posterior distribution. We use this standard algorithm in our experiments for the Bayesian methods.

4 Experiments

4.1 Setup

4.1.1 True functions

The true functions we use to generate the binary classification task lie in the Sobolev space of smoothness 1. We give here the definition the Sobolev space from [18]:

Let $\Omega = [0,1]^d$. For $f : \Omega \to \mathbb{R}$, let $\|f\|_p := (\int_\Omega |f|^p dx)^{\frac{1}{p}}$. Let $|\alpha| = \sum_{j=1}^d |\alpha_j|$ and $D^\alpha f(x) = \frac{\partial^{|\alpha|} f}{\partial^{\alpha_1} x_1 ... \partial^{\alpha_d} x_d}(x)$.

Definition 1.4 (Sobolev space $(W_p^k(\Omega))$). Sobolev space $W_p^k(\Omega)$ with a regularity parameter $k \in \mathbb{N}$ and a parameter $1 \leqslant p \leqslant \infty$ is a set of functions such that the Sobolev norm $\|f\|_{W_p^k} := (\sum_{|\alpha| \leqslant k} \|D^\alpha f\|_p^p)^{\frac{1}{p}}$ is finite.

They correspond to *smooth enough* functions, for which the results from Section 2.1 holds. We squeezed their image through a sigmoid function to get probabilities between 0 and 1. We recall that we are generating an observation $X_0, ..., X_n$ where each X_i takes the value 1 with probability $f(x_i)$, and 0 with probability $1 - f(x_i)$, where x_i is the x-axis point corresponding to the random variable X_i. Figure 6 shows the plot of the true functions g_0, g_1, g_2 and g_3.

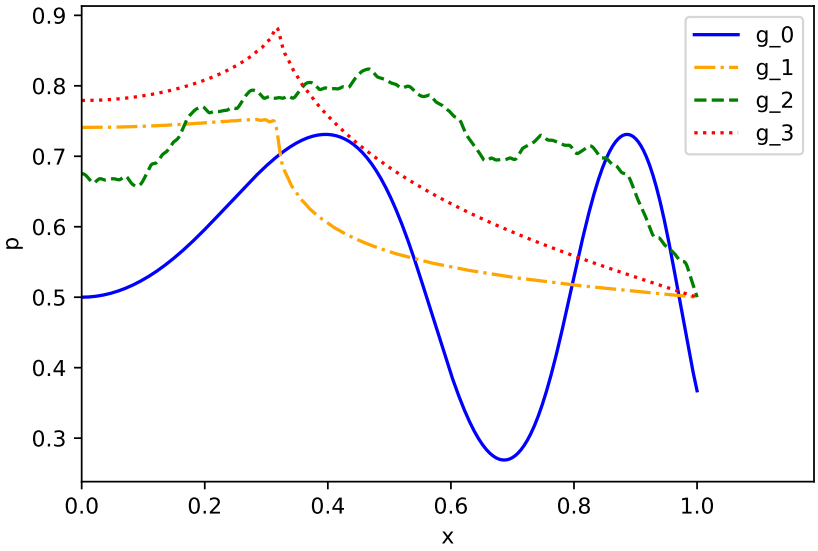


Figure 6: True functions of the binary classification task. The image of each function is passed through a sigmoid function to return probabilities between 0 and 1.

Each function is defined as follow:

$$g_0(x) = sin(10x^2),$$

$$g_1(x) = \sum_{i=1}^{20000} \frac{sin(i)}{i^{(s+0.5)}} cos(\pi x(i-0.5)),$$

$$g_2(x) = \sum_{i=1}^{20000} \frac{sin(i^2)+cos(i^2)}{i^{(s+0.5)}} cos(\pi x(i-0.5)),$$

$$g_3(x) = \sum_{i=1}^{20000} \frac{sin(i)+cos(i)}{i^{(s+0.5)}} cos(\pi x(i-0.5)).$$

4.1.2 Nonparametric setup

We showed in Section 2.1 the theoretical guarantees that these functions can be approximated by our neural network. We recall that the network is fully connected with ReLU activation functions. We choose the following depth and width which guarantees the results from [14]:

$$\text{Depth}(n,s) = \lfloor s * \lfloor \log n \rfloor \rfloor,$$

$$\text{Width}(n,s) = \lfloor n^{\frac{1}{(1+2s)}} \rfloor.$$

Having the structure of the neural network as a variable makes the model nonparametric, as introduced in Section 1.2.3. In the experiments, we will evaluate our method on 15000 and 100000 training points on functions of smoothness 1, which correspond to the network structures $(\text{Width}, \text{Depth}) : (24, 9)$ and $(46, 11)$ respectively.

For all experiments we fix the number of epochs to 100 and the batch size between 36 and 60. The loss function is the binary cross-entropy and is minimized by Adam optimization. The binary classification setup does not allow for ReLU activation function on the last layer, therefore we use the sigmoid function.

4.2 Evaluation

We are interested in the quality of the uncertainty quantification statements on the predictions of a neural network trained on the binary classification task. To do so we evaluate how often the credible intervals generated by the methods cover the true function. We define 3 metrics which are the L-2 distance, L-2 zero distance, and running time to evaluate the results of the experiments

4.2.1 L-2 norm

The L-2 norm of a function is defined by the quantity:

$$\|f\|_2 = \sqrt{\frac{\sum_{i=1}^{n}(f(x_i))^2}{n}}.$$

The L-2 distance between 2 functions is then:

$$d_2(f, g) = \|f - g\|_2 = \sqrt{\frac{\sum_{i=1}^{n}(f(x_i) - g(x_i))^2}{n}}.$$

The metric we use to build credible intervals is $d_2(f, \bar{f})$, where $\bar{f}$ is the average prediction:

$$\bar{f}(x) = \sum_{k=1}^{n_f} \frac{f_k(x)}{n_f}$$

For the bootstrap method, we resample and train neural networks 50 times ($n_f = 50$), while for Bayesian methods we draw 200 vectors from the posterior distribution ($n_f = 200$). We define the 95% credible interval as the interval that contains 95% of the L-2 distances. Its radius is the upper bound of the interval, which is equal to the 95th percentile of the L-2 distances.

We define the L-2 zero distance as the L-2 distance between the true function f_0 and the mean $\bar{f}$: $d_2(f_0, \bar{f})$. The L-2 zero distance is an estimate of the accuracy of the network, while the 95% credible interval is an estimate of how much the predictions deviate from the mean. For each experiment, we are interested in how often the L-2 zero distance is inside the 95% credible interval. In the Empirical Bayesian Neural Network setup, theory motivates a credible ball $B(c_\alpha, R_\alpha)$ blown up by a factor $L = \log n$ to achieve 95% coverage rate.

4.2.2 Number of training points and running time

Comparing the credible intervals on two different sizes of dataset will give information on the behavior of these objects. We can measure for example the contraction rate introduced in Section 1.2.5. We generate 2 different tasks: first, a small task of 15000 training points and then, a large task of 100000 points. We will compare the running time of each method on both tasks. As explained in Section 3.1, we expect the bootstrap method to scale poorly.

5 Results

5.1 Simulation on 15000 training points

Blowup	g_0	g_1	g_2	g_3
None	0%	31.9%	0%	24.7%
$\sqrt{\log n}$	95.9%	100%	98.9%	100%
$\log n$	100%	100%	100%	100%
$(\log n)^3$	100%	100%	100%	100%

Table 1: EBNN 95% credible interval coverage rate of the true function for $n = 15000$ training points (average on 100 experiments).

Table 1 shows the coverage rate of the credible interval generated by the Empirical Bayesian Neural Network for $n = 15000$ training points for 100 experiments. A blow-up of factor $L = \log n$ covers the true function 100% of the time, which is what theory hints for. Table 2 and Table 3 show the coverage rate for the Variational Bayes (ADVI) and bootstrap methods respectively. We see on Table 5 that these two methods have average L2 distances twice as large as for the Empirical Bayesian Neural Network. This explains their better coverage rate. The average running time is sensibly larger for the bootstrap method, 15-fold and 7-fold compared to the Empirical Bayesian Neural Network and the Variational Bayes method respectively. However, Table 4 shows that the bootstrap method has more accuracy than the two others. It comes from the fact that Bayesian methods only train the neural network on half the data, the other half being used for linear regression.

Blowup	g_0	g_1	g_2	g_3
None	80.4%	100%	97.6%	91.9%
$\sqrt{\log n}$	100%	100%	98.9%	100%
$\log n$	100%	100%	100%	100%
$(\log n)^3$	100%	100%	100%	100%

Table 2: ADVI 95% credible interval coverage rate of the true function for $n = 15000$ training points (average on 100 experiments).

Blowup	g_0	g_1	g_2	g_3
None	100%	100%	100%	100%

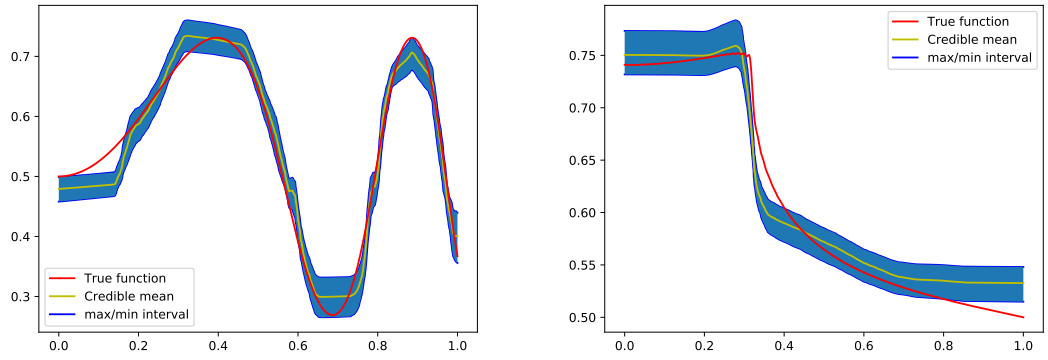
Table 3: Bootstrap 95% min/max interval coverage rate of the true function for $n = 15000$ training points (average on 100 experiments).

Function	g_0	g_1	g_2	g_3	Avg time
EBNN	44.1	25.4	32.6	25.6	4 min
ADVI	46.4	21.7	25.4	34.7	7 min
Bootstrap	21.4	11.3	17.7	12.2	67 min

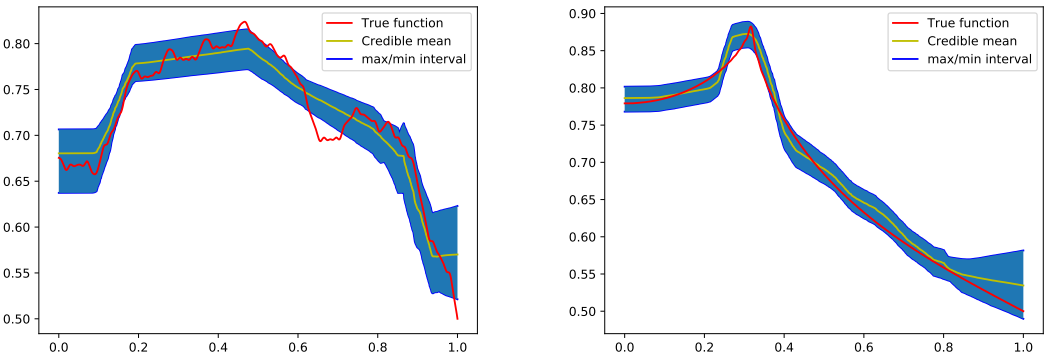
Table 4: Average L2-zero distance (10^{-3}) on 100 experiments for $n = 15000$ training points. The L-2 zero distance $d_2(f_0, \bar{f})$ is a measure of the accuracy of the network.

Function	g_0	g_1	g_2	g_3	Avg time
EBNN	25.9	22.2	19.3	20.8	4 min
ADVI	59.0	50.2	50.5	53.1	9 min
Bootstrap	55.8	27.9	34.5	32.7	67 min

Table 5: Average L2-distances (10^{-3}) on 100 experiments for $n = 15000$ training points.

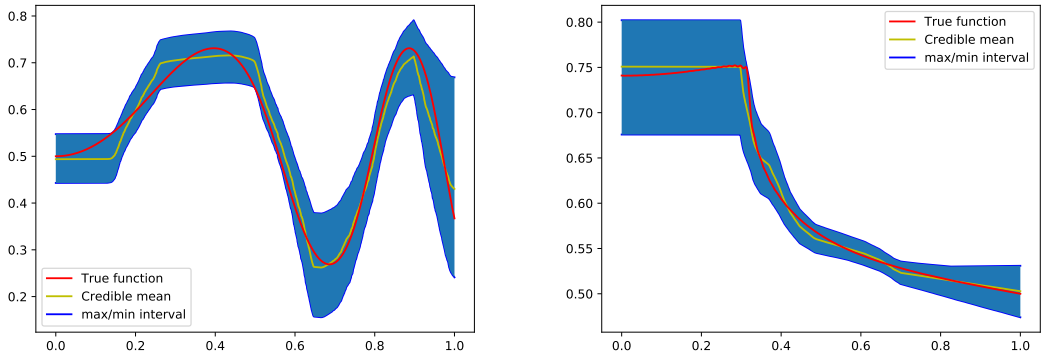


(a) g_0 and g_1

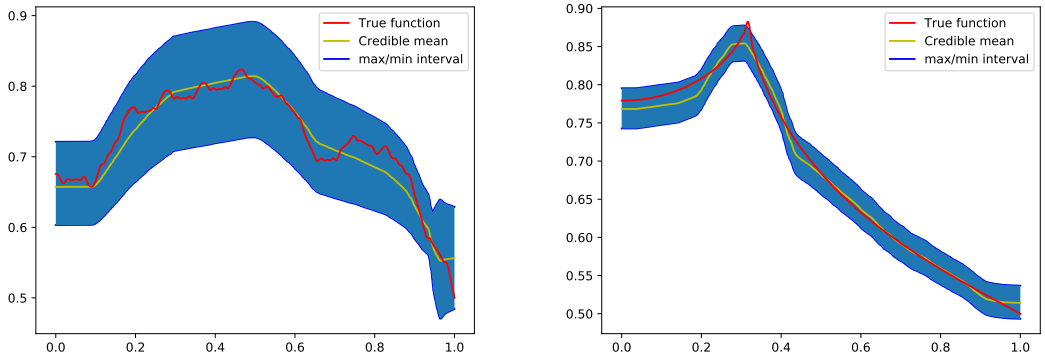


(b) g_2 and g_3

Figure 7: Example of one experiment. Credible mean and max/min interval of the EBNN for $n = 15000$ training points. We see that the 95% credible interval requires a blow-up for coverage, as even the max/min interval does not cover correctly the true function.

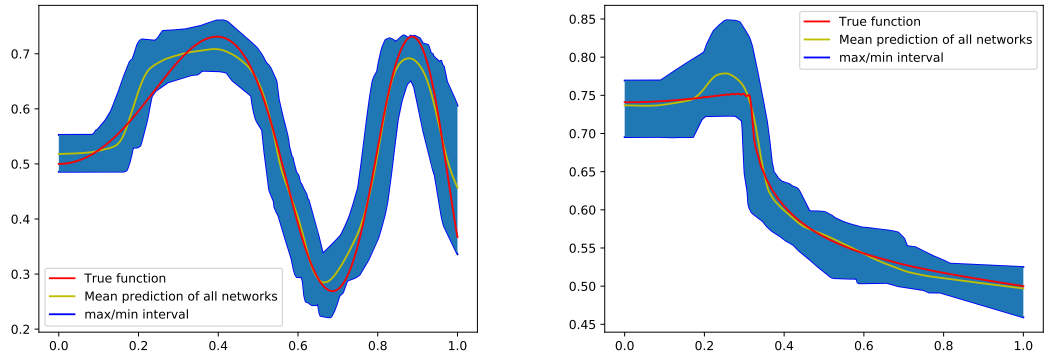


(a) g_0 and g_1

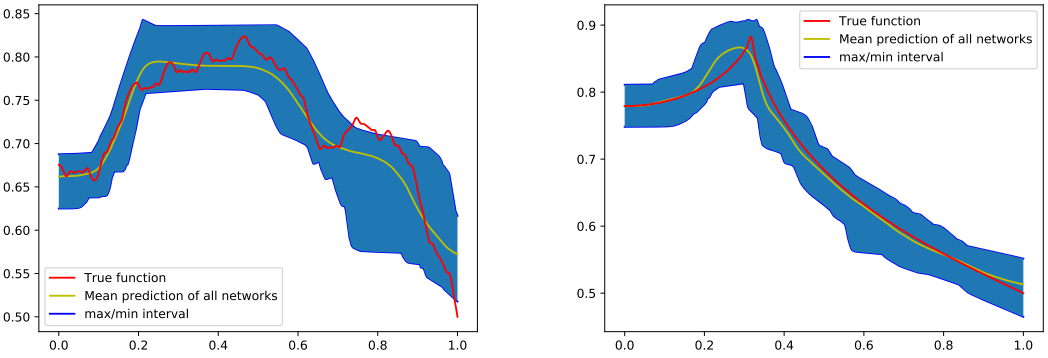


(b) g_2 and g_3

Figure 8: Example of one experiment. Credible mean and max/min interval of the ADVI method for $n = 15000$ training points. The bands generated by ADVI are larger than those generated by the EBNN. The coverage rate is better, but the uncertainty quantification statement might be an overestimate.

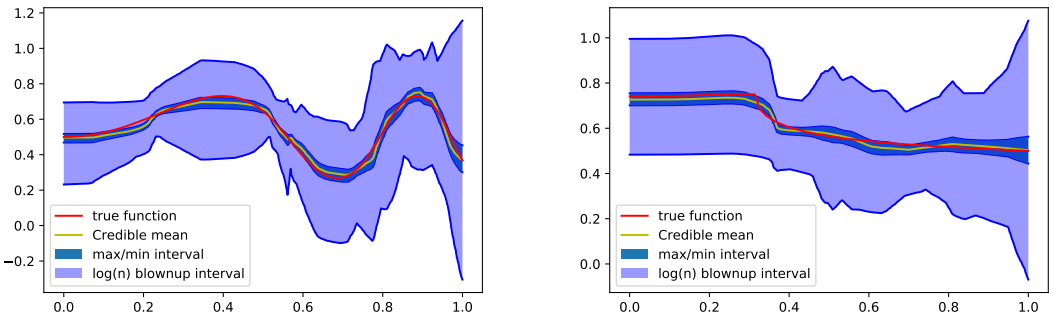


(a) g_0 and g_1

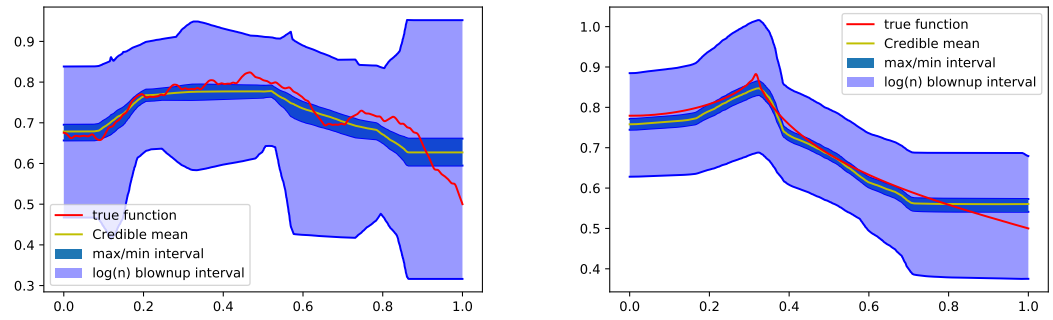


(b) g_2 and g_3

Figure 9: Example of one experiment. Network mean and max/min interval for the bootstrap method for $n = 15000$ training points. The bands are smoother than for the other two methods. They cover the true function in a seemingly accurate way.



(a) g_0 and g_1



(b) g_2 and g_3

Figure 10: Example of one experiment. Credible mean, max/min interval and $\log n$ blow-up interval of the EBNN for $n = 15000$ training points. The blow-up is equal to a factor 9.6. Theory motivates a coverage rate of 95% of the true function given this value of blow-up.

5.2 Simulation on 100000 training points

Blowup	g_0	g_1	g_2	g_3
None	0%	22%	0%	0%
$\sqrt{\log n}$	82%	100%	96%	100%
$\log n$	100%	100%	100%	100%
$(\log n)^3$	100%	100%	100%	100%

Table 6: EBNN 95% credible interval coverage rate of the true function for $n = 100000$ training points (average on 50 experiments).

Table 6 shows that again a blow-up factor of $\log n$ guarantees coverage of the true function. Table 7 follows the earlier ADVI results. Not shown is the coverage rate of the bootstrap method, which

happens to be similar to Table 3. Table 8 shows that Bayesian methods are less accurate in approximating the true function than the bootstrap method, which is mostly due to the fact they don't train on the full dataset. Nevertheless, the Empirical Bayesian Neural Network seems to perform better than the Variational Bayes method in accuracy. Again we see from Table 9 that the Empirical Bayesian Neural Network generates small average L2-distances compared to the other two methods. The scaling for bootstrapping did not go as well as for the Bayesian methods, as it ran on average 20 times longer.

Blowup	g_0	g_1	g_2	g_3
None	46.2%	100%	82.1%	89.2%
$\sqrt{\log n}$	94.9%	100%	100%	100%
$\log n$	100%	100%	100%	100%
$(\log n)^3$	100%	100%	100%	100%

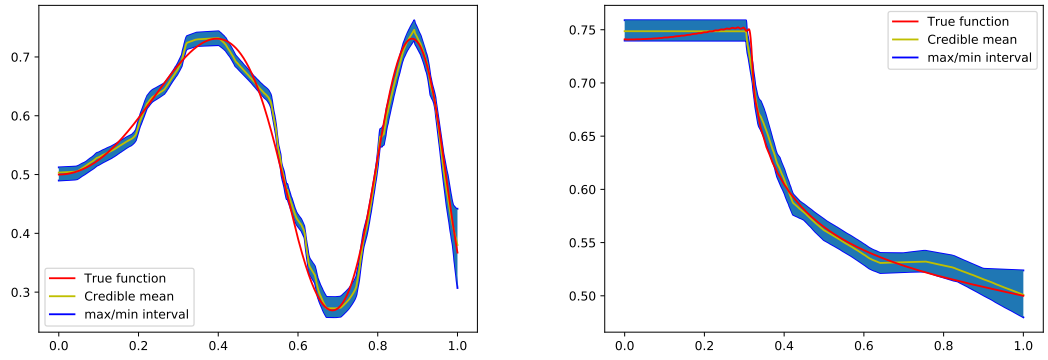
Table 7: ADVI 95% credible interval coverage rate of the true function for $n = 100000$ training points (average on 50 experiments).

Function	g_0	g_1	g_2	g_3	Avg time
EBNN	28.4	11.2	18.5	13.4	14 min
ADVI	65.5	54.9	31.4	18.6	13 min
Bootstrap	10.5	4.8	9.5	6.3	307 min

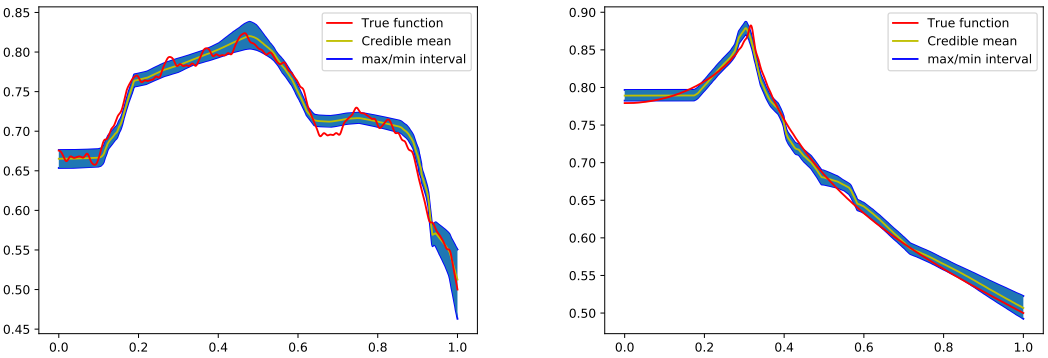
Table 8: Average L2-zero distance (10^{-3}) on 50 experiments for $n = 100000$ training points.

Function	g_0	g_1	g_2	g_3	Avg time
EBNN	10.8	8.9	8.5	8.6	14 min
ADVI	43.2	133.4	50.1	39.6	13 min
Bootstrap	39.9	18.6	23.0	21.3	307 min

Table 9: Average L2-distances (10^{-3}) on 50 experiments for $n = 100000$ training points.

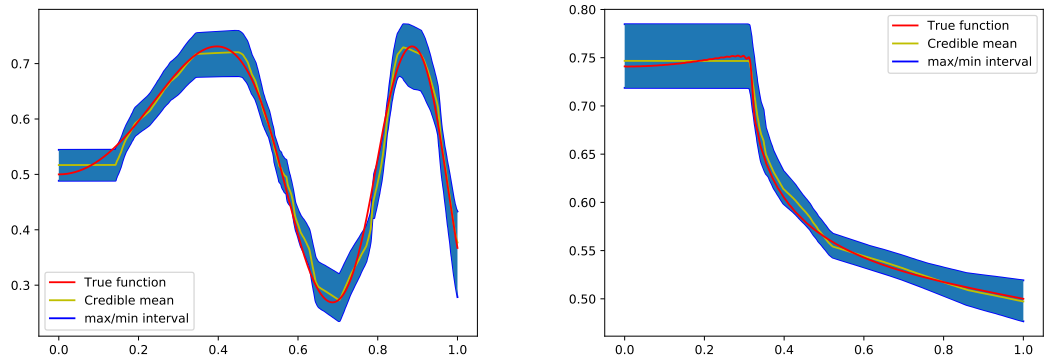


(a) g_0 and g_1

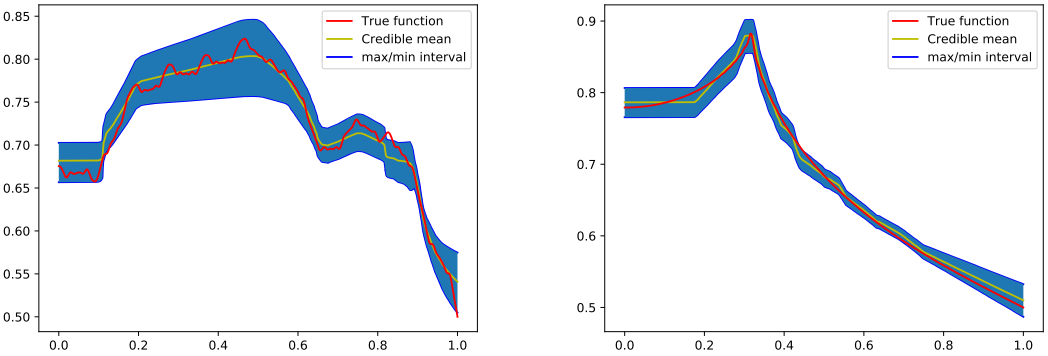


(b) g_2 and g_3

Figure 11: Example of one experiment. Credible mean and max/min interval of the EBNN for $n = 100000$ training points. The interval generated is very narrow compared to the 15000 training points setup. A blow-up is required to cover the true function more consistently.

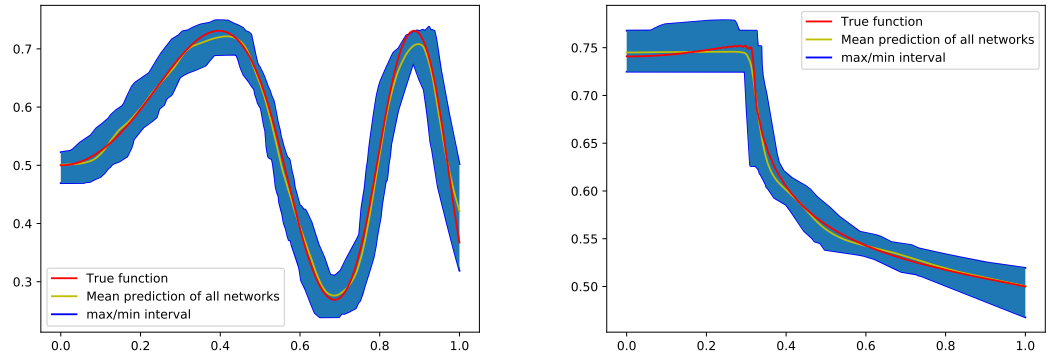


(a) g_0 and g_1

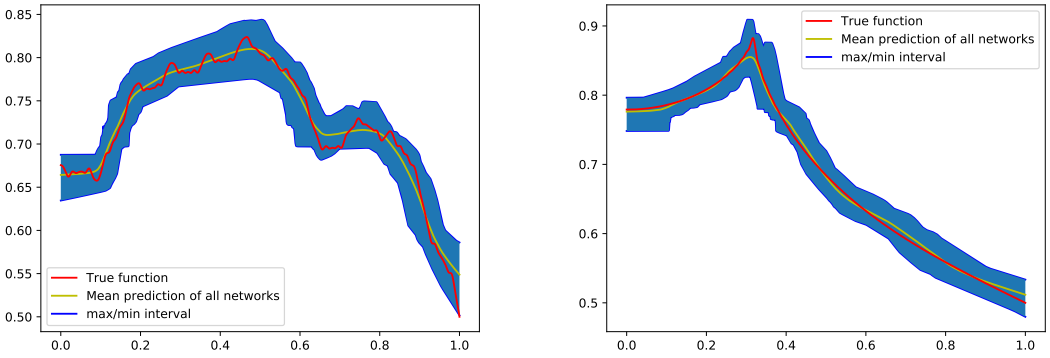


(b) g_2 and g_3

Figure 12: Example of one experiment. Credible mean and max/min interval of the ADVI method for $n = 100000$ training points. Increasing the number of training points has only slightly narrowed the average L2-distances for the ADVI method.

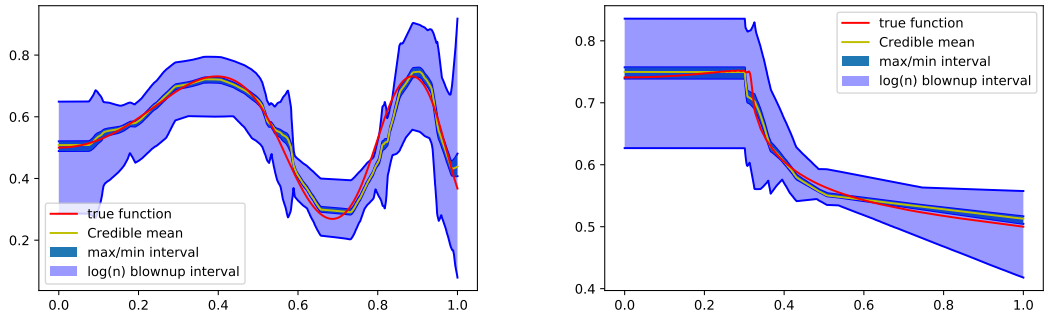


(a) g_0 and g_1

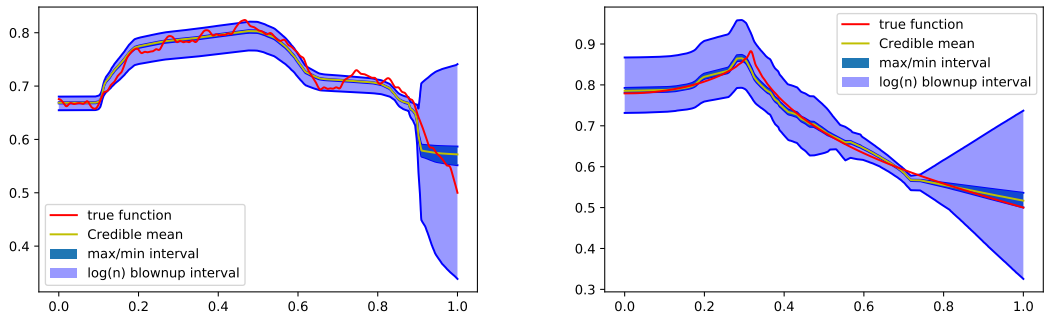


(b) g_2 and g_3

Figure 13: Example of one experiment. Network mean and max/min interval for the bootstrap method for $n = 100000$ training points. Again the bootstrap method generated smoother bands which cover the true function in seemingly optimal way.



(a) g_0 and g_1



(b) g_2 and g_3

Figure 14: Example of one experiment. Credible mean, max/min interval and $\log n$ blow-up interval of the EBNN for $n = 100000$ training points. The blow-up is equal to a factor 11.5. The bands are not as uniform as for the bootstrap method. It seems to highlight zones of higher uncertainty, which might be a feature of more accurate uncertainty statements.

5.3 Summary

We evaluated 3 methods to compute uncertainty quantification statements on a binary classification task. The bootstrap method, which corresponds to training repeatedly neural networks on resample of the data, consistently generated bands that cover the true functions (Figures 9, 13 and Table 3). The width of the bands, evaluated by the average L-2 distances (Tables 5, 9), do not seem to be overestimated and appear more uniformly around the true function than the Bayesian methods. The downside of the method is the running times which are the worst of the three methods by a large margin.

The Empirical Bayesian Neural Network (EBNN), which corresponds to the theory-driven method, generated blown-up credible balls with a 100% coverage rate of the true functions (Tables 1, 6). The bands are uneven around the true functions and appear to highlight zones of higher uncer-

tainty (Figures 10, 14). The scaling is better than the bootstrap method.

The Automatic Differentiation Variational Inference (ADVI) method tends to overestimate the width of the credible bands (Figures 8, 12). There exists no theoretical results that guarantees the uncertainty quantification statements generated by this method.

6 Conclusion

The most straightforward method of uncertainty quantification in neural networks is bootstrapping, which takes advantage of their stochastic nature of learning to explore their variance by training over and over again on a resample of the data. It corresponds quite interestingly to a frequentist approach, in which we draw conclusions using repetition of events. We showed that this method performs well on the binary classification task and generates intervals that cover nicely the true function. However, whether these intervals can be considered uncertainty quantification statements is hard to justify. In essence, they represent variations of the neural network predictions using misrepresentations of the data. And in practice, bootstrapping is meant to be used on linear predictors, which neural networks are not. It means that there are no theoretical guarantees that bootstrapping generates confidence intervals, and therefore uncertainty quantification statements.

How good are uncertainty statements if they cannot be trusted? This problem motivated the creation of the Empirical Bayesian Neural Network which possesses theoretical guarantees. Using the Bayesian framework, this method generates the joint posterior distribution of the weights of the last layer of a neural network, which is then used to calculate the credible intervals with respect to the L2 norm. We are interested in how often these credible intervals cover the true function, and the coverage rate corresponds to our uncertainty quantification statements. We showed that blowing up the intervals by a log factor allows for full coverage of the true function in all our experiments. However, it comes first at a cost of accuracy, because the training data is split in half so the network only trains on a limited sample space, and at a cost of an architecture restriction, which ensures that theoretical guarantees hold.

It is important to note that in practice, we don't know in advance the smoothness of the true function to approximate, and therefore, whether neural networks are effective solutions.

This work highlights the current theoretical understandings of neural networks, and in an effort to make the deep neural networks trustworthy solutions, a simulation study on a theory driven-method to measure the uncertainty of their predictions.

References

[1] Moloud Abdar et al. "A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges". In: *CoRR* abs/2011.06225 (2020). arXiv: 2011.06225. URL: https://arxiv.org/abs/2011.06225.

[2] Sanjeev Arora et al. "A Convergence Analysis of Gradient Descent for Deep Linear Neural Networks". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=SkMQg3C5K7.

[3] Mohamed A. A. Babiker, Mohamed A. O. Elawad, and Azza H. M. Ahmed. "Convolutional Neural Network for a Self-Driving Car in a Virtual Environment". In: *2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*. 2019, pp. 1–6. DOI: 10.1109/ICCCEEE46830.2019.9070826.

[4] Diana Cai. *Schwartz's Theorem and weak posterior consistency*. https://www.dianacai.com/blog/2021/02/14/schwartz-theorem-posterior-consistency/. [Online]. 2021.

[5] Taeryon Choi and Mark J. Schervish. *Posterior consistency in nonparametric regression problems under gaussian processs priors*. URL: http://www.stat.cmu.edu/tr/tr809/tr809.pdf.

[6] S. Franssen. *Frequentist coverage of Empirical Bayesian uncertainty quantification using Deep Neural Network regression*. 2021.

[7] Jakob Gawlikowski et al. *A Survey of Uncertainty in Deep Neural Networks*. 2021. arXiv: 2107.03342 [cs.LG].

[8] Subhashis Ghosal and Aad van der Vaart. *Fundamentals of Nonparametric Bayesian Inference*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2017. DOI: 10.1017/9781139029834.

[9] Lior Hirschfeld et al. "Uncertainty Quantification Using Neural Networks for Molecular Property Prediction". In: *CoRR* abs/2005.10036 (2020). arXiv: 2005.10036. URL: https://arxiv.org/abs/2005.10036.

[10] K. Hornik, M. Stinchcombe, and H. White. "Multilayer Feedforward Networks Are Universal Approximators". In: *Neural Netw.* 2.5 (July 1989), pp. 359–366. ISSN: 0893-6080.

[11] Tom M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2.

[12] Ali Nawaz et al. *Deep Convolutional Neural Network based Classification of Alzheimer's Disease using MRI data*. 2021. arXiv: 2101.02876 [eess.IV].

[13] Nicholas G. Polson and Veronika Ročková. "Posterior Concentration for Sparse Deep Learning". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 2018, pp. 938–949.

[14] Johannes Schmidt-Hieber. "Nonparametric regression using deep neural networks with ReLU activation function". In: *The Annals of Statistics* 48.4 (Aug. 2020). ISSN: 0090-5364. DOI: 10.1214/19-aos1875. URL: http://dx.doi.org/10.1214/19-AOS1875.

[15] Lorraine Schwartz. "On Bayes procedures". In: *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* (1965). DOI: https://doi.org/10.1007/BF00535479.

[16] Leslie N. Smith and Nicholay Topin. "Deep Convolutional Neural Network Design Patterns". In: *CoRR* abs/1611.00847 (2016). arXiv: 1611.00847. URL: http://arxiv.org/abs/1611.00847.

[17] Nicolas Stahl et al. "Evaluation of Uncertainty Quantification in Deep Learning". In: (2020). DOI: https://doi.org/10.1007/978-3-030-50146-4_41. URL: https://link.springer.com/content/pdf/10.1007%2F978-3-030-50146-4_41.pdf.

[18] Taiji Suzuki. "Adaptivity of deep ReLU network for learning in Besov and mixed smooth Besov spaces: optimal rate and curse of dimensionality". In: *arXiv e-prints*, arXiv:1810.08033 (Oct. 2018), arXiv:1810.08033. arXiv: 1810.08033 [stat.ML].

[19] Yuchi Tian et al. “Testing Deep Neural Network based Image Classifiers”. In: *CoRR* abs/1905.07831 (2019). arXiv: 1905.07831. URL: http://arxiv.org/abs/1905.07831.

[20] Matias Valdenegro-Toro. “I Find Your Lack of Uncertainty in Computer Vision Disturbing”. In: *CoRR* abs/2104.08188 (2021). arXiv: 2104.08188. URL: https://arxiv.org/abs/2104.08188.

[21] Li Yang and Abdallah Shami. “On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice”. In: *CoRR* abs/2007.15745 (2020). arXiv: 2007.15745. URL: https://arxiv.org/abs/2007.15745.