A short survey of Bayesian Neural Networks

Fabian Jaeger

July 2023

Abstract

This concise overview introduces Bayesian Neural Networks (BNNs) as a promising alternative to standard neural networks (SNNs), which often suffer from a lack of interpretability, earning them the label of "black box" models. This lack of transparency to a neural network's ability to explain its predictions is especially critical in the sciences and medicine where point estimates are not sufficient and uncertainty estimates are often required. Furthermore, it is argued that incorporating Bayesian concepts into machine learning can also improve the overall accuracy of models. After an introduction and comparison between BNNs and SNNs, in line with the paper from Wilson et. al. [1], we introduce the current understanding of a model's ability to generalize and current shortcomings, such as double descent that plagues standard neural networks, and how certain BNN frameworks prove to be a promising candidate in alleviating some of those pitfalls. However, BNNs still suffer from practical implementations due to their inferior scalability to larger datasets and models. Some of the most common approximations to the optimization of training and estimation of Bayesian statistical quantities are then introduced.

Contents

1 Bayesian Neural Networks

Modern deep-learning methods are incredibly powerful tools that can tackle a wide range of challenging problems. However, due to their nature as black boxes, quantifying the uncertainty associated with their predictions can be difficult [2]. Bayesian statistics provide a formal framework to comprehend and measure the uncertainty inherent in deep neural network predictions.

Bayesian Neural Networks incorporate the principles of Bayesian Inference, which can be particularly useful when dealing with limited data or when making predictions in uncertain environments. BNNs extend traditional neural networks by introducing probabilistic weights and biases.

BNNs train the model weights as probability distributions rather than searching for a singular optimal value as with SNN. This makes them more robust and allows them to generalize better with less overfitting. The process of finding these distributions associated with the weights is called *marginalization*.

One standard approach to finding point estimates of weights in standard neural networks would be Maximum Likelihood Estimators (MLE) while for BNN, the estimate would be rather Maximum A Posteriori (MAP) or predictive distribution. While SNNs would use differentiation to find the optimal value through gradient descent, BNNs rely on Markov Chain Monte Carlo (MCMC), Variational Inference, and Normalizing Flows [3], given the presence of hard-to-compute integrals which has the unfortunate consequence of increased computational complexity.

2 Advantages of BNNs

2.1 Sources of uncertainty

Usually, in SNN applications, our data $\mathcal{D}$ is merely a sample of the process we are modeling. In these cases, we are looking for models that generalize to unseen data. Therefore, it is useful to express the uncertainty about our model due to the lack of data. This uncertainty is commonly referred to as epistemic uncertainty or parameter uncertainty.

BNNs address epistemic uncertainty, as explained above, by treating the model's weights as probabilistic variables rather than fixed parameters. This allows the model to capture the uncertainty in its own parameters and produce probabilistic predictions, indicating the model's confidence in its predictions.

Usually, there is another source of uncertainty called aleatoric uncertainty, which originates directly from the process that we are modeling. This uncertainty, also known as data uncertainty, arises from the inherent variability in the data itself and is the noise in the labels that cannot be explained by the inputs. Often, this uncertainty is called "irreducible noise". Bayesian Neural Networks play a crucial role in quantifying and managing also aleatoric uncertainty by incorporating uncertainty estimates in the output of the model.

2.2 Bayesian Inference

In Bayesian Inference, given for example some known input-output pairs in a supervised learning setting, we seek to compute the posterior $p(\boldsymbol{\theta}|\mathcal{D})$, which is the conditional distribution of the parameters of a model given the training data $\mathcal{D}$. The posterior can be determined using Bayes' theorem:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})}{p(\mathcal{D})} \propto p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta}) \tag{1}$$

where $p(\mathcal{D}|\boldsymbol{\theta})$ represents the likelihood and $p(\boldsymbol{\theta})$ the prior. Our interest in the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ lies in the fact that we want to compute the *posterior predictive distribution* to model unseen data

$$p(\boldsymbol{y}|\boldsymbol{x}, \mathcal{D}) = \int p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta} = \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\left[p(\boldsymbol{y}|\boldsymbol{x}, \mathcal{D})\right]. \tag{2}$$

Equation (2) represents a *Bayesian Model Average* (BMA) as it averages the predictions of all plausible models weighted by

their posterior. This yields a nice interpretation of the predictive distribution as an infinite ensemble of networks, hence *ensemble learning*, which can represent many hypothesis models we believe to be possible. This is in net contrast to MLE methods in SNN where only one set of parameters $\boldsymbol{\theta}^*$, therefore one hypothesized model, is used for predictions.

$$\begin{array}{ccc} \text{SNN} & & \text{BNN} \\ \boldsymbol{\theta}^* & \Rightarrow & p(\boldsymbol{\theta}|\mathcal{D}) \\ y_{\boldsymbol{\theta}^*}(x) & \Rightarrow & p(\boldsymbol{y}_{\boldsymbol{\theta}}|\boldsymbol{x}, \mathcal{D}) \end{array}$$

An immediate benefit of specifying distributions over the model parameters and predictions is that we can quantify our uncertainty about these things, e.g., by computing their variance. This is especially relevant when learning from small datasets; standard neural net training will overfit for the reasons discussed above, but Bayesian Inference will find the best explanation for the model parameters given the available data, which typically have high uncertainty when data is scarce. In the limit of dataset size far exceeding the number of neurons, the inferred distributions sharpen and begin to resemble the solutions from the standard training; for modern networks we don't typically reach this limit in practice, so having a notion of model uncertainty is helpful.

The classical training of an SNN can be viewed as a limit of the Bayesian Inference in the case when $p(\boldsymbol{\theta}|\mathcal{D}) \approx \delta(\boldsymbol{\theta} = \boldsymbol{\theta}^*)$, where the delta function is zero everywhere except at $\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D})$. The difference between a classical and Bayesian approach will depend on how sharp the posterior becomes. If the posterior is sharply peaked and the predictive distribution does not vary much where the posterior has mass, then there may be no difference between the two approaches. However in modern SNN problems the likelihood $p(\mathcal{D}|\boldsymbol{\theta})$ has wide support, not favoring a specific set of parameters. In such scenarios, it is advantageous to employ BNNs.

In Bayesian Machine learning, the prior $p(\boldsymbol{\theta})$ is often imposed. Common choices include a pre-determined class, such as Gaussian priors $p(\boldsymbol{\theta} = \mathcal{N}(\alpha, \beta)$ or a more systematic approach trying to estimate the parameters of the prior from the data with the help of empirical Bayes approach.

Two computationally difficult integrals arise during the Bayesian setup which limits its use in practical applications. The first one arises from the computation of the marginal likelihood $p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$, while the second corresponds to the posterior predictive distribution (2). It is possible to estimate these two integrals via various algorithms which can be categorized as either sampling-based or variational.

Sampling algorithms are mostly based on Markov Chain Monte Carlo (MCMC) methods, the most relevant in BNN is the Metropolis-Hastings algorithm. For example, we can compute $p(y|x, \mathcal{D})$ by sampling a finite set of parameters $\{\theta_1, \theta_2, ..., \theta_N\}$, whose distribution matches $p(\boldsymbol{\theta}|\mathcal{D})$ in the limit for large N to compute the predictive distribution. When the size of the model grows, however, sampling algorithms become less efficient, and variational algorithms are preferred when computing intractable distributions like the posterior.

Variational methods directly model the posterior using a parametrized distribution $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$, then iteratively improve the approximation via an optimization problem. The measure of closeness that is commonly used is the Kullback-Leibler divergence shown in equation (24)[4]. The process of learning the parameters of the distribution closely resembles SNN training and is called stochastic variational inference.

2.3 Generalization

Typically, in traditional machine learning, a model's ability to generalize effectively (low generalization error) is linked to a small difference between training and test errors. However, a seminal paper by Zhang. et. al. [5] revealed that even large neural networks with very small training and test errors can fit random labels in the training data, highlighting a lack of understanding in quantifying a model's generalization capacity. Moreover, in the following section, we will discuss a peculiar phenomenon known as Double Descent[6], which further accentuates this observation.

In the paper by Wilson et. al. ([1]) it is argued that the random labeling is not surprising if analyzed through a Bayesian framework with the help of the concepts of support and inductive bias. Inductive bias refers to the inherent assumptions or preferences a machine learning model acquires from its training data, guiding it to favor particular solutions over others and enabling it to generalize well to new data. This bias plays a crucial role in helping the model make sensible decisions even in the presence of noisy or limited training data. To achieve good inductive bias, various techniques are commonly employed, including regularization, dropout, early stopping, and data augmentation, among others. Additionally, the selection of an appropriate model architecture that strikes a balance between complexity and representational capacity is of utmost importance.

Models with fewer parameters tend to possess a stronger inductive bias, primarily because they are constrained to fit a specific class of data. However, this limitation may also result in insufficient generalization to other types of data, as the model's support might not be extensive enough. In this context, support refers to the range of dataset classes that a model can effectively accommodate, corresponding to situations where the likelihood $p(\mathcal{D}|\mathcal{M}) > 0$.

Thus from a statistical perspective, we want a large support and specific inductive bias of a model to generalize well, as will become evident in the next subsection.

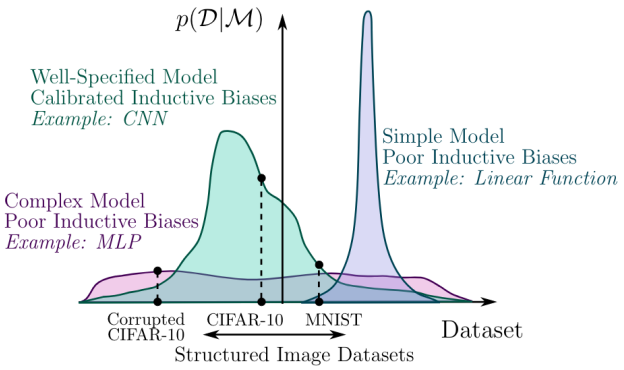


Figure 1: The vertical axis represents the marginal likelihood or Bayesian evidence and represents how good a mode is at fitting a specific dataset. Figure taken from reference [1]

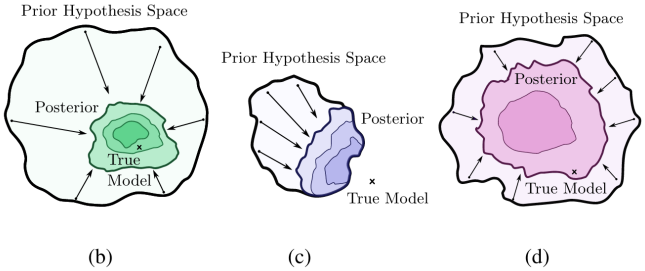


Figure 2: Prior hypothesis space and posterior for (b) CNN (c) linear model (d) MLP.

2.3.1 Generalization for BNNs

To exemplify the delicate balance between a wide support and a strong inductive bias, consider the illustration presented in Figure (1) from Wilson et. al. [1]. The blue linear model displays truncated support, as it lacks the flexibility to capture quadratic behaviors in the data. Consequently, the marginal likelihood's probability mass is sharply concentrated around datasets with linear behaviors.

In contrast, a traditional MLP (multi-layer perceptron) exhibits a support that is distributed too evenly across all classes of datasets. As a result, the model lacks the necessary inductive bias to effectively capture and generalize well on specific image datasets.

The ideal model, illustrated here as a CNN (convolutional neural network), strikes a balance between casting a wide support and maintaining a strong inductive bias to learn well on image data. It successfully learns the distinctive features of the dataset, enabling improved generalization on diverse classes of data (different image sets).

In the context of a Bayesian framework, our objective is for the posterior $p(\mathcal{D}|\mathcal{M})$ to converge towards the correct solution while adhering to the appropriate inductive bias. For this purpose, the prior hypothesis space, also known as support, must be sufficiently large to encompass the true model, as depicted in Figure (2). If the inductive bias aligns with the true model, and the true model lies within the prior hypothesis space (scenario (b) in the Figure, akin to the CNN case in Figure (2)), the model will contract around the accurate solution.

However, if the prior hypothesis space is too restricted, limiting the model's flexibility, it may not include the correct solution, even if the inductive bias is accurate. Consequently, the posterior will contract towards an incorrect solution, as illustrated by the example of the linear function.

Furthermore, when the support is excessively broad, and the model lacks the appropriate inductive bias, the model will not effectively contract (depicted in Figure (d), representing the MLP example). In such cases, the model's ability to converge to the correct solution is impaired.

In practical terms, our choice of a model involves finding a balance between flexibility and support. We aim to select a highly flexible model that can cover a wide range of possibilities (large support). However, to ensure sensible inductive biases, we need to complement this flexibility with a suitable choice of prior $p(\boldsymbol{\theta})$ on the coefficients that govern the distribution over functions[1] $p(f(\boldsymbol{x}))$.

2.4 Double Descent

Traditional neural networks suffer from what is referred to as double descent, as depicted in Figure (3). In the classical regime, governed by the Bias-Variance Tradeoff, as model complexity increases, the expected test error initially decreases, reaches a minimum, and then starts increasing due to overfitting. However, double descent challenges this conventional understanding and introduces a new perspective. It reveals that, with sufficiently large datasets, there exists a second descent in the error curve, referred to as the modern interpolating regime, occurring after the initial peak of overfitting. This unexpected behavior suggests that complex models can still generalize well and achieve lower test errors, even after the conventional overfitting point.

By employing Bayesian model averaging, models with suitable posterior approximation and sensible priors, it was demonstrated in [1] that the unexpected double descent behavior observed in traditional approaches does not manifest in Bayesian deep learning models. Instead, we observe the traditionally expected monotonic behavior of improved accuracy with increasing model flexibility. This will be further elaborated in section (4.6).

[1] To better understand this concept, consider again the simple linear model where we have $f(x) = \theta_0 + \theta_1 x$. The parameters θ_0 and θ_1 are generated according to our prior $p(\boldsymbol{\theta})$ with $\boldsymbol{\theta} = (\theta_0, \theta_1)$. Sampling from this prior thus gives us a distribution over linear functions with different slopes and intercepts.

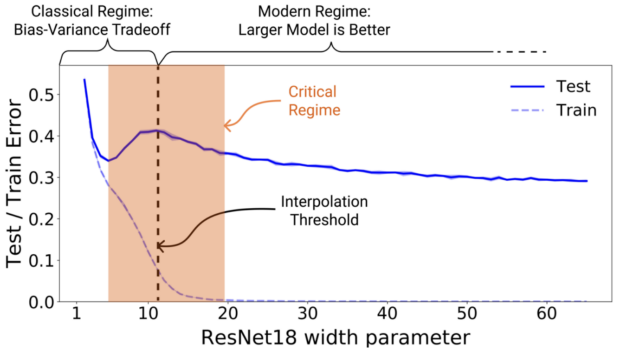


Figure 3: Figure taken from [6]. It shows the general behaviour of the test and train error as a function of the model flexibility (encapsulated in the width of the ResNet-18 architecture here) where double descent appears.

3 General optimization procedure

In a standard machine learning optimization process for a supervised problem, given a dataset $\mathcal{D} = \left\{\left(\boldsymbol{x}_n \in \mathbb{R}^N, \boldsymbol{y}_n \in \mathbb{R}^M\right)\right\}$, where $\boldsymbol{y}_n \in \mathbb{R}^M$ for regression and $\boldsymbol{y}_n \in 0, 1^M$ for binary classification, we typically perform Maximum Likelihood Estimation. The objective is to find the optimal parameters $\boldsymbol{\theta} \in \mathbb{R}^P$, which correspond to the weights and biases in a typical multi-layer perceptron for example. This optimization process is formulated as follows:

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmax}}\, p(\mathcal{D}|\boldsymbol{\theta}) \tag{3}$$

The optimization strategy is frequently reformulated as a minimization problem[7]. To further mitigate overfitting, a penalty term $C(\boldsymbol{\theta})$ (regularizer) is commonly incorporated. This modified minimization strategy is known as empirical risk minimization and can be expressed as follows:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}}\ [-\log p(\mathcal{D}|\boldsymbol{\theta}) + \lambda C(\boldsymbol{\theta})] \tag{4}$$

where $\lambda \geq 0$ serves as a tunable regularization parameter.

In the frequentist framework, we consider $\boldsymbol{\theta}$ as a fixed but unknown parameter that solely depends on the data. In contrast, the Bayesian framework treats all unknowns as random variables. Consequently, we hold prior beliefs about the distribution $p(\boldsymbol{\theta})$, and upon observing the data, we update these beliefs using the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{x})$. The Maximum A Posterior (MAP) framework aligns more closely with Bayesian principles, as it aims to maximize the posterior expression (1)

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmax}}\, p(\boldsymbol{\theta}|\mathcal{D}) \propto \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmax}}\, p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta}). \tag{5}$$

If we consider i.i.d samples the likelihood factorizes into conditional probabilities

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p\left(\boldsymbol{y}_n|\boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{\theta})\right) \tag{6}$$

where we introduce the feature function $\boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{x})$ with $\boldsymbol{\theta} \in \mathbb{R}^P$, which predicts the outputs of the output distribution (e.g., a

Gaussian). Reformulating as a minimization problem using log-loss, we obtain the following optimization strategy:

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{\text{MAP}} &= \underset{\boldsymbol{\theta}\in\Theta}{\text{argmin}} \left(-\log p(\boldsymbol{\theta}) + \sum_{i=1}^{N} \ell_n(\boldsymbol{x}_n, \boldsymbol{y}_n, \boldsymbol{\theta}) \right) \\ &= \underset{\boldsymbol{\theta}\in\Theta}{\text{argmin}}\, \mathcal{L}(\boldsymbol{x}_n, \boldsymbol{y}_n, \boldsymbol{\theta})\end{aligned} \tag{7}$$

where we define the empirical loss ℓ_n and log joint distribution: $\mathcal{L}$ as follows:

$$\begin{aligned}\ell_n(\boldsymbol{x}_n, \boldsymbol{y}_n, \boldsymbol{\theta}) &= -\log p\left(\boldsymbol{y}_n | \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{\theta})\right) \\ \mathcal{L}(\boldsymbol{x}_n, \boldsymbol{y}_n, \boldsymbol{\theta}) &= -\left[\sum_{n=1}^{N} \log p\left(\boldsymbol{y}_n | \boldsymbol{f}\left(\boldsymbol{x}_n, \boldsymbol{\theta}\right)\right) + \log p(\boldsymbol{\theta}) \right]\end{aligned} \tag{8}$$

It is worth noting that the regularized MLE estimate can be obtained by considering the regularized empirical risk problem and assuming that the complexity parameter $C(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta})$ and $\lambda = 1$.

4 Posterior Approximations

4.1 Maximum A Posterior estimation

The simplest approximate inference method involves computing the Maximum A Posterior (MAP) estimate:

$$\hat{\boldsymbol{\theta}} = \text{argmax}\, p(\boldsymbol{\theta}|\mathcal{D}) = \text{argmax} \log p(\boldsymbol{\theta}) + \log p(\mathcal{D}|\boldsymbol{\theta}). \tag{9}$$

Here, we assume that the posterior concentrates all its probability mass on a single value:

$$p(\boldsymbol{\theta}|\mathcal{D}) \approx \delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}), \tag{10}$$

which corresponds to a typical machine learning approach where a single set of optimal parameters is chosen. However, such a procedure has several drawbacks, including a lack of uncertainty quantification for the parameters and limited utilization of Bayesian methods. In a Bayesian framework, we have the opportunity to use other model selection processes that have been shown to improve accuracy.

To capture the uncertainty effectively, we can employ various inference algorithms, including but not limited to the Grid approximation, variational inference, Markov Chain Monte Carlo (MCMC), and the Laplace approximation. These methods allow for a better characterization of parameter uncertainty and enable the full potential of Bayesian approaches.

4.2 Laplace Approximation

To provide further insight, we reframe the posterior expression (1) as follows:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})}{p(\mathcal{D})} = \frac{p(\mathcal{D}, \boldsymbol{\theta})}{p(\mathcal{D})}, \tag{11}$$

where, in the second step, we use the property that the conditional probability can be expressed as the quotient of the joint probability with respect to the marginal. Within the Laplace Approximation[8], we assume that we can represent this posterior as a multivariate Gaussian:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{1}{Z} e^{-\mathcal{E}(\boldsymbol{\theta})} \quad \text{with} \quad \mathcal{E}(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta}, \mathcal{D}), \tag{12}$$

where $\mathcal{E}(\boldsymbol{\theta})$ (corresponding to the loss function in the MAP estimate) is referred to as the energy function. The normalization constant, $Z = p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\theta$, serves as an analogy to the canonical ensemble in statistical physics.

By performing a Taylor expansion of the posterior approximation around the lowest energy state $\hat{\boldsymbol{\theta}}$ (obtained, for example, using any standard machine learning optimization with regularized empirical risk minimization or the simple MAP approach explained above), we arrive at the following expression:

$$\begin{aligned}\mathcal{E}(\boldsymbol{\theta}) \simeq\ & \mathcal{E}(\hat{\boldsymbol{\theta}}_{\text{MAP}}) + (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_{\text{MAP}})^T \boldsymbol{\nabla}_\theta \mathcal{E}(\boldsymbol{\theta}) \\ & + \frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_{\text{MAP}})^T (\boldsymbol{\nabla}^2_\theta \mathcal{E}(\boldsymbol{\theta}) \mid_{\hat{\boldsymbol{\theta}}_{\text{MAP}}})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_{\text{MAP}})\end{aligned} \tag{13}$$

The first term vanishes at the mode $\hat{\boldsymbol{\theta}}_{\text{MAP}}$ since the gradient is zero. Consequently, we obtain the joint probability distribution:

$$\begin{aligned}\hat{p}(\mathcal{D}, \boldsymbol{\theta}) \simeq\ & e^{-\mathcal{E}(\hat{\boldsymbol{\theta}})} \\ & \times \exp\left[-\frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_{\text{MAP}})^T (\boldsymbol{\nabla}^2_\theta \mathcal{E}(\boldsymbol{\theta}) \mid_{\hat{\boldsymbol{\theta}}_{\text{MAP}}})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_{\text{MAP}}) \right]\end{aligned} \tag{14}$$

which, by comparing (11) and (12), yields the following posterior approximation:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{1}{Z}\hat{p}(\boldsymbol{\theta}, \mathcal{D}) \simeq \mathcal{N}(\boldsymbol{\theta} \mid \hat{\boldsymbol{\theta}}_{\text{MAP}}, \boldsymbol{\Sigma}), \tag{15}$$

where

$$\boldsymbol{\Sigma} = \left(\boldsymbol{\nabla}^2_\theta \mathcal{E}(\boldsymbol{\theta}) \mid_{\hat{\boldsymbol{\theta}}_{\text{MAP}}} \right)^{-1}, \tag{16}$$

and $\boldsymbol{\Sigma}$ is the inverse of the Hessian.

The normalizing constant Z corresponding to the marginal likelihood or evidence can be determined by the general formulation of a multivariate Gaussian and is given by

$$Z \simeq e^{-\mathcal{E}(\boldsymbol{\theta})} (2\pi)^{D/2} (\det \boldsymbol{\Sigma})^{1/2} \tag{17}$$

To explicitly compute the Hessian, we need to evaluate it for the joint log expression $\mathcal{L}$ from equation (8). For now, we will neglect the prior term $p(\boldsymbol{\theta})$ and focus on the log-likelihood term for each data point. The first and second derivatives are as follows:

$$\begin{aligned}\boldsymbol{\nabla}_{\boldsymbol{\theta}} \log p(\boldsymbol{y} \mid \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})) &= \mathbf{J}_{\boldsymbol{\theta}}(\boldsymbol{x})^\top \boldsymbol{r}(\boldsymbol{y}; \boldsymbol{f}) \\ \boldsymbol{\nabla}^2_{\boldsymbol{\theta}} \log p(\boldsymbol{y} \mid \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})) &= \mathbf{H}_{\boldsymbol{\theta}}(\boldsymbol{x})^\top \boldsymbol{r}(\boldsymbol{y}; \boldsymbol{f}) \\ & \quad - \mathbf{J}_{\boldsymbol{\theta}}(\boldsymbol{x})^\top \boldsymbol{\Lambda}(\boldsymbol{y}; \boldsymbol{f}) \mathbf{J}_{\boldsymbol{\theta}}(\boldsymbol{\theta})\end{aligned} \tag{18}$$

where $\mathbf{J}_{\boldsymbol{\theta}}(\boldsymbol{x})_{ci} = \partial f_c(\boldsymbol{\theta}, \boldsymbol{x})/\partial\theta_i \in \mathbb{R}^{M\times P}$ is the Jacobian (recall M is the dimension of the output and P the dimension of the parameters $\boldsymbol{\theta}$), $[\mathbf{H}_{\boldsymbol{\theta}}(\mathbf{x})]_{cij} = \frac{\partial^2 f_c(\mathbf{x}, \boldsymbol{\theta})}{\partial\theta_i \partial\theta_j} \in \mathbb{R}^{C\times M\times M}$ the Hermitian, $\mathbf{r}(\boldsymbol{y}; \boldsymbol{f}) = \boldsymbol{\nabla}_{\boldsymbol{f}} \log p(\boldsymbol{y}|\boldsymbol{f})$ the residual and $\boldsymbol{\Lambda}(\boldsymbol{y}; \boldsymbol{f}) = -\boldsymbol{\nabla}^2_{\boldsymbol{f}\boldsymbol{f}} \log p(\boldsymbol{y}|\boldsymbol{f})$ the per noise-input term.

In general, when considering the prior, we have the flexibility to choose any twice-differentiable log-density prior $p(\boldsymbol{\theta})$. However, even for a simple case like the weight-decay regularizer $C(\boldsymbol{\theta}) = \frac{1}{2}\gamma^{-2}|\theta|^2$, where we assume a zero-mean Gaussian prior $\mathcal{N}(\theta|0, \gamma^2 I)$, scalability issues arise. For a supervised learning setting, the Hessian at the MAP estimate is given by:

$$\boldsymbol{\nabla}^2_{\boldsymbol{\theta}} \mathcal{E}(\boldsymbol{\theta}) \mid_{\hat{\boldsymbol{\theta}}_{\text{MAP}}} = -\gamma^{-2} I - \sum_{n=1}^{N} \boldsymbol{\nabla}^2_{\boldsymbol{\theta}} \log p(y_n | \boldsymbol{f}(x_n, \boldsymbol{\theta})) \mid_{\hat{\boldsymbol{\theta}}_{\text{MAP}}} \tag{19}$$

The second term in this equation scales quadratically with the number of network parameters, making it computationally infeasible for large networks with millions of parameters. Additionally, for most neural networks, the log-likelihood is not convex, leading to a non-positive definite Hessian. Consequently, we encounter two significant challenges: 1) the optimization problem is non-convex, making it difficult to find a suitable global minimum, and 2) the optimization procedure becomes more unstable in general.

4.2.1 Gauss-Newton approximation

One way to address the scalability issues is to perform a Gauss-Newton approximation to the Hessian. This involves neglecting the Hermitian term in (18), effectively linearizing the equation:

$$\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{y} \mid \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})) \simeq \mathbf{J}_{\boldsymbol{\theta}}(\boldsymbol{x})^\top \boldsymbol{\Lambda}(\boldsymbol{y}; \boldsymbol{f}) \mathbf{J}_{\boldsymbol{\theta}}(\boldsymbol{x}), \tag{20}$$

where $\mathbf{J}_{\boldsymbol{\theta}}(\boldsymbol{x})$ is the Jacobian matrix and $\boldsymbol{\Lambda}(\boldsymbol{y}; \boldsymbol{f})$ is a diagonal matrix containing the second derivatives of the log-likelihood with respect to the output predictions. This approximation also ensures that the optimization remains positive semi-definite. Additionally, computing Jacobians involves only first-order derivatives, which are efficiently implemented in many modern machine learning tools.

For the common choice of a Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\beta})$, the Laplace approximation simplifies to:

$$p(\boldsymbol{\theta}|\mathcal{D}) \approx \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\Sigma}_{\text{GGN}}), \tag{21}$$

where

$$\boldsymbol{\Sigma}^{-1}_{\text{GGN}} = \sum_{n=1}^{N} \mathbf{J}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_n\right)^\top \boldsymbol{\Lambda}\left(\boldsymbol{y}_n, \boldsymbol{f}_n\right) \mathbf{J}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_n\right) + \boldsymbol{\beta}^{-1}. \tag{22}$$

However, inverting this matrix still takes $\mathcal{O}(P^3)$.Therefore, additional approximation schemes are necessary for such cases. One simple approach is diagonal factorization, which ignores off-diagonal terms in the matrix.

In summary, the Laplace approximation is favored for its simplicity as an inference approximation scheme. Apart from computing the MAP estimate, which can be accomplished using traditional optimization algorithms, the main computational requirement is to calculate the inverse of the Hessian matrix at $\boldsymbol{\theta}_{\text{MAP}}$. This straightforward implementation makes it suitable for obtaining statistical measures of pre-trained models and has been conveniently implemented in torch by Immer et. al. [9] in the `Laplace Redux` library. Moreover, the Laplace approximation provides a reasonable approximation for models with moderate parameter dimensions.

However, challenges arise when applying the Laplace approximation to deep machine learning models with an extensive number of parameters. In such cases, the computation of the inverted Hessian becomes computationally expensive due to the high dimensionality of the problem. Consequently, for very large neural networks, the Laplace approximation may become impractical, and alternative methods such as Monte Carlo sampling techniques like Markov Chain Monte Carlo or variational inference might be preferred to handle the complexity of parameter uncertainty estimation.

4.3 Variational Inference

Methods like Monte Carlo Markov Chain algorithms have proven to be effective for sampling from exact posteriors. However, they also encounter scalability issues, primarily because it may require a large number of iterations until the algorithm converges to the stationary distribution (i.e., the desired posterior). This slow convergence can make MCMC computationally expensive and impractical for large-scale problems.

Alternatively, a popular method is Variational Inference, also known as variational Bayes. In this approach, we propose an approximate distribution (referred to as the variational distribution) that belongs to a tractable family of distributions $\mathcal{Q}$ parameterized by a set of parameters $\boldsymbol{\phi}$. The goal is to find the best approximate solution to the intractable posterior $p(\boldsymbol{\theta}|\mathcal{D})$:

$$q^* = \underset{q \in \mathcal{Q}}{\operatorname{argmin}}; D(q||p) \tag{23}$$

where $D(q||p)$ is typically the Kullback-Leibler Divergence between the exact posterior and the variational distribution, defined as:

$$D_{\text{KL}}(q|p) = \int_{\boldsymbol{\theta}'} q(\boldsymbol{\theta}'|\boldsymbol{\phi}) \log\left(\frac{q(\boldsymbol{\theta}'|\boldsymbol{\phi})}{p(\boldsymbol{\theta}'|\mathcal{D})}\right) d\boldsymbol{\theta}' \tag{24}$$

The above expression quantifies the discrepancy between the approximate and exact posterior distributions. Instead of directly minimizing the divergence, a more convenient quantity, known as the Evidence Lower Bound (ELBO), can be used, which is related to the KL-Divergence. The ELBO is defined as:

$$\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\phi})}[\log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta}|\boldsymbol{\psi})] \tag{25}$$

By maximizing the ELBO, we effectively minimize the KL-Divergence, thereby making the variational distribution closer to the true posterior. Since the ELBO involves an expectation with respect to the variational distribution, it still requires some calculation of the often intractable posterior $p(\boldsymbol{\theta}|\mathcal{D})$.

The most common choice for the posterior approximation is to use multivariate Gaussians, which take the form $q(\boldsymbol{\theta}|\boldsymbol{\phi}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$. In this case, the parameters $\boldsymbol{\phi}$ consist of the mean $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$ of the Gaussian distribution. This approach differs from the Laplace approximation, where the posterior covariance was derived from the Hessian at the MAP estimate (as shown in equations 15 and 16).

The variational Approximation offers a more global representation of the posterior, but it still faces potential challenges related to locating suboptimal minima and being limited to specific classes of functions $q \in \mathcal{Q}$, similar to the constraints of the Laplace approximation. Notably, the use of Gaussian distributions for the approximate posterior might restrict its expressiveness, unlike methods like MCMC, which do not impose such assumptions and can explore more diverse posteriors[10].

4.4 Deep Ensembles

To address this, other approaches like Deep Ensembles (see the section (4.4)) have been proposed, which involve training multiple models with different initialization to capture the uncertainty and explore different regions of the parameter space. These ensembles can provide a more robust and comprehensive representation of the often multi-modal posterior distribution, as illustrated in Figure 4. In this ensemble-based approach, the models are equally weighted by a sum of Dirac delta functions, with each function allocating mass to the MAP mode of a specific model m:

$$p(\boldsymbol{\theta}|\mathcal{D}) \approx \frac{1}{M} \sum_{m=1}^{M} \delta\left(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^m_{\text{MAP}}\right) \tag{26}$$

Here, M represents the number of models in the ensemble, and $\hat{\boldsymbol{\theta}}^m_{\text{MAP}}$ denotes the MAP estimate for model m.

4.4.1 Multi-SWAG

Multi-SWAG ([1]) builds upon the Deep Ensemble concept and adds an extra layer of complexity by fitting a Gaussian to each of the local modes, often referred to as basins of attractions, represented by the MAP estimates $\hat{\boldsymbol{\theta}}^m_{\text{MAP}}$. This is achieved using the SWAG (Stochastic Weight Averaging Gaussian) approximation, which provides a method to fit these Gaussians (though we will not delve into its details here). As a result, the posterior distribution is approximated as a mixture of Gaussian components:

$$p(\boldsymbol{\theta}|\mathcal{D}) \approx \frac{1}{M} \sum_{m=1}^{M} \mathcal{N}\left(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}_m, \boldsymbol{\Sigma}_m\right) \tag{27}$$

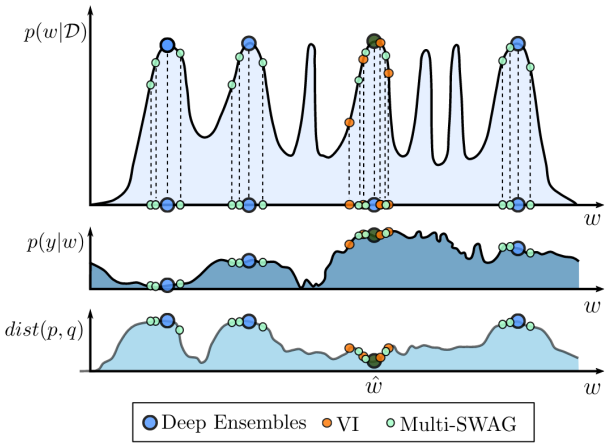


Figure 4: Figure extracted from [1]. The label w refers to the model parameters $\boldsymbol{\theta}$ in our case. The top-most figure represents the true posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ with representations of this posterior from variational inference (orange), deep ensembles (blue), and Multi-SWAG (green). The middle plot represents the conditional predictive distribution $p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})$. The top and middle part thus make up the components of the posterior predictive distribution (2). The bottom plot represents the distance between the true posterior predictive distribution p and the approximation q.

The key advantage over simple Deep Ensembles lies in the ability to directly generate new posterior samples at each mode, thanks to the mixture of Gaussians representation. This approach provides a more fine-grained approximation of the posterior distribution, allowing for better exploration of the parameter space and a more comprehensive understanding of model uncertainty.

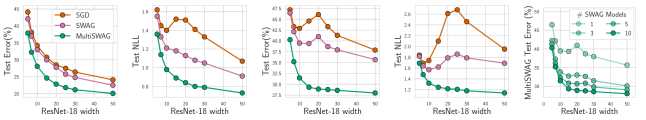
4.5 Comparison of posterior approximations

To compare the computation of the predictive distribution using different methods, namely deep ensembles, variational inference for a single basin, and Multi-SWAG, Wilson et. al. [1] employed an illustrative example depicted in Figure 4. Recall that the posterior predictive distribution is the integral of the conditional predictive and the posterior both of which are shown in the top and middle plot respectively. As such we want to factor of the two to be as large as possible.

Variational inference selects samples from a single basin of attraction in the posterior $p(\boldsymbol{\theta}|\mathcal{D})$, corresponding to parameters $\boldsymbol{\theta}$ where the Kullback-Leibler divergence between the true posterior and the approximate posterior is minimized, as depicted in the bottom plot. On the other hand, Deep Ensembles considers multiple basins of attraction due to being an ensemble of models with different setups, each potentially having a different MAP estimate $\hat{\boldsymbol{\theta}}^m_{\text{MAP}}$. Understanding how the conditional predictive distribution $p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})$ varies with respect to the parameters (as seen in the middle plot) reveals the benefits of sampling from different basins.

Although the probability mass for basins of attraction other than the one chosen by variational inference might be lower, their large posterior predictions can still have a substantial impact on the overall posterior predictive distribution. Consequently, when samples are obtained near $\hat{\boldsymbol{\theta}}$ ($\hat{\omega}$ in the plot), Deep Ensembles and Multi-SWAG show significant improvements in the posterior predictive compared to variational inference by sampling from additional modes in different basins.

To summarize: having multiple basins of attraction, obtained by distinct components of the ensemble, leads to greater functional diversity compared to Bayesian approaches that primarily concentrate on approximating the posterior within a single basin of attraction.



(a) True Labels (Err) (b) True Labels (NLL) (c) Corrupted (Err) (d) Corrupted (NLL) (e) Corrupted (# Models)

Figure 5: Figure taken from [1]. Plot (a) the test error and (b) test negative log-likelihood for the uncorrupted dataset with respect to the ResNet-18 width. In (c) and (d) we find the test error and NLL for a partially corrupted dataset. In plot (e) we find the test error for Multi-SWAG with a varying number of SWAG models for a the corrupted dataset. We observe that with an increasing number of SWAG models we reduce the double descent behaviour monotonically.

4.6 Back to Double Descent

With the insights gained from the previous section, we can now understand the claim made by Wilson et al. [1] that Bayesian neural networks do not exhibit double descent behavior. This boils down to their ability (for certain approximations as we will see) to perform an exhaustive *multimodal* Bayesian model average [1]. To examine this hypothesis further, they conducted experiments using Multi-SWAG, SWAG, and SGD with ResNet-18 models of varying widths. The results for error and negative log-likelihood (NLL) are illustrated in Figure 5. These experiments were also conducted on a partially corrupted dataset, where 20% of the labels were randomly reshuffled (Figure (e) and (d)).

The error plot (c) clearly shows the traditional double descent behavior for SWAG (red curve). However, MultiSWAG effectively mitigates the issue of double descent by implementing a comprehensive multimodal Bayesian model averaging approach. As a result, the error curve scales monotonically with increasing model flexibility, demonstrating the model's ability to adapt to different complexities while avoiding the double descent phenomenon.

References

1. Wilson, A. G. & Izmailov, P. *Bayesian Deep Learning and a Probabilistic Perspective of Generalization* Mar. 30, 2022. arXiv: 2002.08791[cs,stat]. http://arxiv.org/abs/2002.08791 (2023).
2. Bykov, K. *et al. Explaining Bayesian Neural Networks* Aug. 23, 2021. arXiv: 2108.10346[cs,stat]. http://arxiv.org/abs/2108.10346 (2023).
3. Winter, S., Campbell, T., Lin, L., Srivastava, S. & Dunson, D. B. *Machine Learning and the Future of Bayesian Computation* Apr. 21, 2023. arXiv: 2304.11251[cs,stat]. http://arxiv.org/abs/2304.11251 (2023).
4. Kullback, S. & Leibler, R. A. On Information and Sufficiency. *The Annals of Mathematical Statistics* **22,** 79–86. ISSN: 0003-4851. http://projecteuclid.org/euclid.aoms/1177729694 (2023) (Mar. 1951).
5. Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. *Understanding deep learning requires rethinking generalization* Feb. 26, 2017. arXiv: 1611.03530[cs]. http://arxiv.org/abs/1611.03530 (2023).
6. Nakkiran, P. *et al. Deep Double Descent: Where Bigger Models and More Data Hurt* Dec. 4, 2019. arXiv: 1912.02292[cs,stat]. http://arxiv.org/abs/1912.02292 (2023).

7. Murphy, K. P. *Probabilistic machine learning: an introduction* 826 pp. ISBN: 978-0-262-04682-4 (The MIT Press, Cambridge, Massachusetts, 2022).

8. Daxberger, E. *et al. Laplace Redux – Effortless Bayesian Deep Learning* Mar. 14, 2022. arXiv: 2106.14806[cs, stat]. http://arxiv.org/abs/2106.14806 (2023).

9. Immer, A., Korzepa, M. & Bauer, M. Improving predictions of Bayesian neural nets via local linearization.

10. Jospin, L. V., Buntine, W., Boussaid, F., Laga, H. & Bennamoun, M. Hands-on Bayesian Neural Networks – a Tutorial for Deep Learning Users. *IEEE Computational Intelligence Magazine* **17,** 29–48. ISSN: 1556-603X, 1556-6048. arXiv: 2007.06823[cs,stat]. http://arxiv.org/abs/2007.06823 (2023) (May 2022).