Bayesian deep learning: A review

Written in August 2020

Apostolos Psaros

Contents

1 Introduction

From the introduction of Kendall and Gal (2017): In May 2016 there was the first fatality from an assisted driving system, caused by the perception system confusing the white side of a trailer for bright sky. In a second recent example, an image classification system erroneously identified two African Americans as gorillas, raising concerns of racial discrimination. If both of these algorithms were able to assign a *high level of uncertainty to their erroneous predictions*, then the system may have been able to make better decisions and likely avoid disaster.

In these notes the motivation for and some basic notions related to Bayesian deep learning are briefly reviewed. In Section 1, Bayesian hierarchical modeling is introduced and standard training of neural networks (NNs) arises as a special case. Specifically, following exactly the Bayesian framework involves taking into account many compelling hypotheses that explain our data, whereas the standard training of NNs is an approximation that bases its results on a single hypothesis. In this regard, Bayesian neural networks (BNNs) facilitate accurate data analysis with the right amount of confidence on the output results. Further, in Section 2 the basic components of the Bayesian framework are briefly reviewed; i.e., model building, parameter inference, and model criticism.

1.1 Bayesian hierarchical modeling

1.1.1 Bayesian model average

Given a set of paired datapoints $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ we are interested in obtaining the value of y^* at some new location x^* by constructing a model $\mathcal{H}$; i.e., we are interested in $p(y^*|x^*, \mathcal{D}, \mathcal{H})$. If the model has parameters $\boldsymbol{w} = [w_1, \dots, w_k]^T$ to be inferred from the data, then we integrate over the posterior density of the parameters given the data, i.e.,

$$p(y^*|x^*, \mathcal{D}, \mathcal{H}) = \int p(y^*|x^*, \boldsymbol{w}, \mathcal{H}) p(\boldsymbol{w}|\mathcal{D}, \mathcal{H}) d\boldsymbol{w} \tag{1}$$

With $p(y^*|x^*, \boldsymbol{w}, \mathcal{H})$ it is assumed that the value y^* will be given as the deterministic output of our model, say $f_{\mathcal{H}}(x^*; \boldsymbol{w})$, contaminated by some noise. In these notes the terms "parameters" and "weights" are used interchangeably. These terms are distinguished from the term "hyperparameters" introduced later.

Next, for a Gaussian white noise model we obtain

$$p(y^*|x^*, \boldsymbol{w}, \mathcal{H}) = \frac{1}{\sqrt{2\pi\sigma_\epsilon^2}} \exp\left(-\frac{(y^* - f_{\mathcal{H}}(x^*; \boldsymbol{w}))^2}{2\sigma_\epsilon^2}\right) \tag{2}$$

If no noise in the output is considered, Eq. (1) becomes

$$p(y^*|x^*, \mathcal{D}, \mathcal{H}) = \int \delta(y^* - f_{\mathcal{H}}(x^*; \boldsymbol{w}))p(\boldsymbol{w}|\mathcal{D}, \mathcal{H})d\boldsymbol{w} \tag{3}$$

Note that Eq. (3) is still a probability density. If a single value for y^* is required, denoted by $\hat{y}$, then, according to decision theory (e.g., Bishop, 2006 p. 46), for minimizing the expected squared loss error, $\hat{y}$ should be the conditional mean $\mathbb{E}[y^*|x^*, \mathcal{D}, \mathcal{H}]$ given by

$$\hat{y} = \mathbb{E}[y^*|x^*, \mathcal{D}, \mathcal{H}] = \int\int y^* \delta(y^* - f_{\mathcal{H}}(x^*; \boldsymbol{w}))p(\boldsymbol{w}|\mathcal{D}, \mathcal{H})d\boldsymbol{w}dy^* \tag{4}$$

and simplified as

$$\hat{y} = \int f_{\mathcal{H}}(x^*; \boldsymbol{w})p(\boldsymbol{w}|\mathcal{D}, \mathcal{H})d\boldsymbol{w} \tag{5}$$

Note that even if a single output value $\hat{y}$ is required, its determination involves an *integration over all plausible values* of $\boldsymbol{w}$ given (conditioned on) the data; i.e., by admitting and incorporating model uncertainty, also known as epistemic uncertainty (see Gal, 2016 and Kendall and Gal, 2017).

Eqs. (1), (3) and (5) are all examples of a Bayesian model average (BMA). As Wilson and Izmailov (2020) puts it, these are not controversial equations, but simply the sum and product rules of probability. Rather than betting everything on just one hypothesis (single setting of parameters), we utilize all settings of parameters, weighted by their posterior probabilities. Since multiple values of $\boldsymbol{w}$ are considered, we have an *ensemble of models* (instead of a single model) defined by the posterior $p(\boldsymbol{w}|\mathcal{D}, \mathcal{H})$ given via Bayes rule as

$$p(\boldsymbol{w}|\mathcal{D}, \mathcal{H}) = \frac{p(\mathcal{D}|\boldsymbol{w}, \mathcal{H})p(\boldsymbol{w}|\mathcal{H})}{p(\mathcal{D}|\mathcal{H})} \tag{6}$$

In Eq. (6), $p(\mathcal{D}|\boldsymbol{w}, \mathcal{H})$ is the likelihood of $\mathcal{D}$ given a particular setting of parameters $\boldsymbol{w}$ and the noise model of Eq. (2); $p(\boldsymbol{w}|\mathcal{H})$ is the prior probability of the parameters $\boldsymbol{w}$ as defined by the model $\mathcal{H}$; and $p(\mathcal{D}|\mathcal{H})$ is called marginal likelihood or evidence because it represents the probability that out of all possible datasets modeled by $\mathcal{H}$, it happens that we "see" $\mathcal{D}$. The evidence $p(\mathcal{D}|\mathcal{H})$ is given as

$$p(\mathcal{D}|\mathcal{H}) = \int p(\mathcal{D}|\boldsymbol{w}, \mathcal{H})p(\boldsymbol{w}|\mathcal{H})d\boldsymbol{w} \tag{7}$$

i.e., given random samples drawn from the prior $p(\boldsymbol{w}|\mathcal{H})$ and then used in conjuction with the noise model of Eq. (2), $p(\mathcal{D}|\mathcal{H})$ represents the probability that the dataset $\mathcal{D}$ is produced.

1.1.2 Maximum a posteriori (MAP) estimate

If in Eq. (5) we consider the approximation

$$p(\boldsymbol{w}|\mathcal{D},\mathcal{H}) \approx \delta(\boldsymbol{w}-\hat{\boldsymbol{w}}) \tag{8}$$

where $\hat{\boldsymbol{w}}$ is the maximum value of the posterior given as

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\text{argmax}}\; p(\boldsymbol{w}|\mathcal{D},\mathcal{H}) \tag{9}$$

or equivalently as

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\text{argmax}}\;\; \log p(\mathcal{D}|\boldsymbol{w},\mathcal{H}) + \log p(\boldsymbol{w}|\mathcal{H}) \tag{10}$$

then Eq. (5) becomes

$$\hat{y} \approx f_{\mathcal{H}}(x^*;\hat{\boldsymbol{w}}) \tag{11}$$

Historical note: Recall that the typical training of NNs involves minimizing the sum of an additive loss function between the data and the model outputs, e.g.,

$$\mathcal{L}_{\mathcal{D}} = \sum_{i=1}^{N}(y_i - f_{\mathcal{H}}(x_i;\boldsymbol{w}))^2 \tag{12}$$

and of a regularization term, e.g.,

$$\mathcal{L}_{\boldsymbol{w}} = \sum_{j=1}^{k} w_j^2 \tag{13}$$

i.e., the optimal setting of the network parameters are obtained as

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\text{argmin}}\; \mathcal{L}(\boldsymbol{w}) \tag{14}$$

where $\mathcal{L}(\boldsymbol{w}) = \alpha\mathcal{L}_{\boldsymbol{w}} + \beta\mathcal{L}_{\mathcal{D}}$ is the training loss, and α, β are black-box parameters to be tuned (typically the ratio α/β is tuned; see also MacKay, 1995b p. 19 for a comment on this). In this regard, Tishby et al. (1989) introduced a probabilistic view of learning and assigned a meaning to the parameters of Eq. (14). Specifically, in Tishby et al. (1989) it is shown that if the minimization of the loss $\mathcal{L}_{\mathcal{D}}$ is to be expressed as a maximum likelihood problem, then the likelihood $p(\mathcal{D}|\boldsymbol{w},\mathcal{H})$ *must be written as*

$$p(\mathcal{D}|\boldsymbol{w},\mathcal{H}) = \frac{1}{Z(\beta)}\exp(-\beta\mathcal{L}_{\mathcal{D}}) \tag{15}$$

where $Z(\beta)$ is a normalization constant. Further, for the quadratic error function of Eq. (12) the consistency condition of Eq. (15) dictates the use of the Gaussian noise model of Eq. (2) with $\beta = 1/\sigma_\epsilon^2$. Therefore, the only statistical interpretation of minimizing the NN Euclidean loss is maximizing a Gaussian likelihood over the network outputs. Overall, the standard training of NNs of Eq. (14) is equivalent to the MAP estimate of Eq. (10) and the parameters α and β can be viewed as the scale parameters of the prior and of the noise model, respectively.

Finally, note that although the MAP estimate involves a posterior and a prior, it is not Bayesian because it bets everything on a single hypothesis; whether we average (marginalize over the parameters) is what makes a technique Bayesian.

1.1.3 Hyperparameters and hyperpriors

According to the Bayesian framework (Box and Tiao, 1973) all nuisance parameters should be integrated over, as we did in Eq. (1) with $\boldsymbol{w}$. As a result, since the prior and the noise model include their own parameters (typically referred to as hyperparameters), we should integrate over α and β as well. Thus, Eq. (6) becomes

$$p(\boldsymbol{w}|\mathcal{D},\mathcal{H}) = \int p(\boldsymbol{w}|\mathcal{D},\alpha,\beta,\mathcal{H})p(\alpha,\beta|\mathcal{D},\mathcal{H})d\alpha d\beta \tag{16}$$

where

$$p(\boldsymbol{w}|\mathcal{D},\alpha,\beta,\mathcal{H}) = \frac{p(\mathcal{D}|\boldsymbol{w},\beta,\mathcal{H})p(\boldsymbol{w}|\alpha,\mathcal{H})}{p(\mathcal{D}|\alpha,\beta,\mathcal{H})} \tag{17}$$

and

$$p(\alpha,\beta|\mathcal{D},\mathcal{H}) = \frac{p(\mathcal{D}|\alpha,\beta,\mathcal{H})p(\alpha,\beta|\mathcal{H})}{p(\mathcal{D}|\mathcal{H})} \tag{18}$$

In Eq. (18), $p(\alpha,\beta|\mathcal{H})$ is known as hyperprior. Therefore, Eq. (1) becomes

$$p(y^*|x^*,\mathcal{D},\mathcal{H}) = \int p(y^*|x^*,\boldsymbol{w},\beta,\mathcal{H})p(\boldsymbol{w},\alpha,\beta|\mathcal{D},\mathcal{H})d\alpha d\beta d\boldsymbol{w} \tag{19}$$

If we also consider different models $\mathcal{H}$, then we should *integrate over all plausible models* as well and obtain $p(y^*|x^*,\mathcal{D})$, where we condition only on the data and the location x^* we are interested in.

Clearly, exact integration in Eq. (16) (or in Eq. (19)) for real problems is infeasible and thus, approximation techniques are typically utilized (see MacKay, 1994 for an interesting discussion).

1.2 Motivation for Bayesian neural networks

Two main reasons for using and adapting the Bayesian framework in NN training are calibration and accuracy. Excellent discussions are provided by Gal (2016) and Wilson and Izmailov (2020).

1.2.1 Calibration in the classification setting

Calibration refers to the confidence of the NN on the output. In classification NNs, without even using Bayesian techniques, the output provides both a class prediction and its confidence. A classification NN is calibrated if its accuracy is the same as its confidence, i.e., for binary classification

$$P(Y = 1 | f_{\mathcal{H}}(X) = p) = p, \text{for all } p \in [0, 1] \tag{20}$$

where X, Y are drawn from the data-generating distribution. Guo et al. (2017) has shown that modern classification NNs are *miscalibrated* (overconfident); i.e., the class probabilities (levels of confidence) the NNs produce do not correspond to actual probabilities. In other words, an output vector $[0.8, 0.15, 0.05]$ does not mean that in 100 predictions with confidence 0.8, 80 will be correctly classified. Guo et al. (2017) claims that modern NNs, because of the large number of layers and their large widths, exchange calibration for accuracy. In this regard, Guo et al. (2017) reviews some methods for measuring calibration and for post-hoc recalibration, briefly discussed in Sections 1.2.1.1-1.2.1.3 (can be skipped at first reading).

1.2.1.1 Reliability diagram: The output confidence takes values from 0 to 1. In this regard, the confidence axis $[0, 1]$ is divided into M interval bins of size $1/M$. Consider, next, a set of NN predictions with different confidence levels. These predictions are assigned to the corresponding bins I_m using an index set B_m. Further, the average accuracy and confidence of each bin are expressed as

$$acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i - y_i) \tag{21}$$

and

$$conf(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i \tag{22}$$

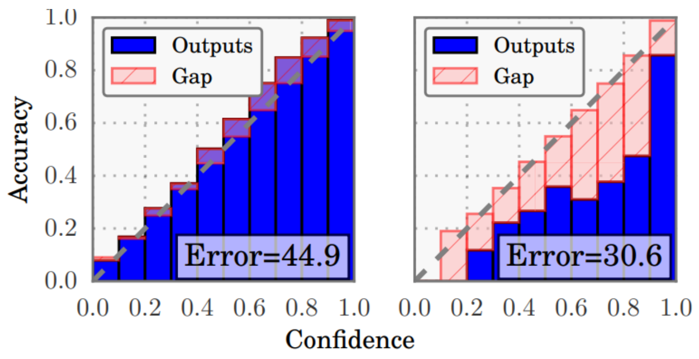


Fig. 1. Figure 1 from Guo et al. (2017): Confidence vs accuracy for a 5-layer LeNet (left) and a 110-layer ResNet (right). Gap shows miscalibration. For example, on the right hand side, we see that on average, an 80% output confidence does not correspond to 80% accuracy, but less.

where y_i and $\hat{y}_i$ denote the true and predicted labels, and $\hat{p}_i$ the confidence on the predicted class. Then, confidence and accuracy can be plotted as in Fig. 1, where the gap $|acc(B_m) - conf(B_m)|$ represents miscalibration in the m-th interval bin.

1.2.1.2 Other metrics: The expected calibration error is a weighted average of the calibration gaps $|acc(B_m) - conf(B_m)|$ over the interval bins and the maximum calibration error is the maximum value of $|acc(B_m) - conf(B_m)|$. The negative log likelihood on unseen data is also a metric of calibration.

1.2.1.3 Post-hoc recalibration: Consider a binary classification NN and a calibration set of size n used for recalibration of the NN (this set could be the same as the validation set used for tuning hyperparameters). One way to recalibrate the predicted confidence is to divide the confidence axis into M interval bins and assign each prediction $\hat{p}_i$ into the corresponding bin. Then a calibrated score θ_m is assigned to each bin. In other words, if $\hat{p}_i$ is assigned to bin m then the corresponding prediction is no longer $\hat{p}_i$, but $\hat{q}_i = \theta_m$. Next, consider that M and the bin boundaries a_m for every m are fixed. The optimal values of the calibrated scores θ_m can be obtained by minimizing the bin-wise squared loss, i.e.,

$$\min_{\theta_1,\ldots,\theta_m} \sum_{m=1}^{M} \sum_{i=1}^{n} \mathbf{1}(a_m \le \hat{p}_i < a_{m+1})(\theta_m - y_i)^2 \tag{23}$$

This is called histogram binning. Further, isotonic regression pertains to optimizing also M and the bin boundaries.

Another approach is Platt scaling, according to which the recalibrated predictions become $\hat{q}_i = \sigma(az_i+b)$, where z_i is the nonprobabilistic prediction of the classifier (before it is passed through the sigmoid function) and a, b are parameters to be optimized using the negative log likelihood loss over the calibration set (for fixed weights).

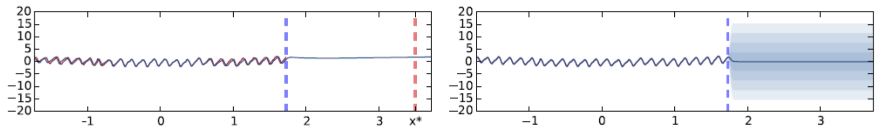
An equivalent way to view the above techniques, as described in Kuleshov et al. (2018), is as approximators of the accuracy for given values of confidence; i.e., as approximators of $P(Y = 1|f_{\mathcal{H}}(X) = p)$ in Eq. (20). Given a value of confidence $\hat{p}_i$ from the NN, the recalibrator (trained on the calibration set) provides an estimate of the accuracy $P(Y = 1|f_{\mathcal{H}}(X) = \hat{p}_i)$; this becomes the final output $\hat{q}_i$. Following the recalibration step, the NN is *on average calibrated*.

The above techniques can be extended to multi-class classification (Guo et al., 2017). An interesting extension of Platt scaling uses a single "temperature" parameter $T > 0$ for all classes. In this regard, the confidence prediction becomes

$$\hat{q}_i = \max_k \sigma_{SM}(\boldsymbol{z}_i/T)^{(k)} \tag{24}$$

where $\boldsymbol{z}_i$ is the logit vector and σ_{SM} denotes the softmax function. The optimal T is obtained with respect to the negative log likelihood loss over the calibration set. Also, temperature scaling does not change the maximum of the softmax function, i.e., the predicted class will be the same, only the confidence changes. Therefore, temperature scaling does not affect the model accuracy. Guo et al. (2017) notes that for remedying miscalibration, temperature scaling is the simplest, fastest, and most straightforward of the methods they considered, and surprisingly is often the most effective. Motivated by the results of Guo et al. (2017), Wenzel et al. (2020) and Wilson and Izmailov (2020) further studied the effects of temperature scaling.

1.2.1.4 Calibration via epistemic uncertainty: As noted in Ovadia et al. (2019), in practice, once a model is deployed the distribution over observed data may shift and eventually be very different from the original training data distribution. In that case, we would like the predictions to indicate that a model "knows what it does not know" due to the inputs straying away from the training data distribution (see Section 2.4). In this regard, Ovadia et al. (2019) has found that post-hoc calibration with temperature scaling leads to well-calibrated uncertainty on the independent and identically distributed (i.i.d.) test set and for small values of dataset shift, but is significantly outperformed by methods that take epistemic uncertainty into account as the



(a) Standard deep learning model (b) Probabilistic model

Fig. 2. Figure 1.2 from Gal (2016): Standard NN vs a probabilistic model trained on data to the left of the blue dashed line. Different shades of blue represent half a standard deviation. Marked with a dashed red line is a point far away from the data: standard deep learning models confidently predict an unreasonable value for the point; the probabilistic model predicts an unreasonable value as well but with the additional information that the model is uncertain about its prediction.

shift increases.

In general, miscalibration happens because we ignore epistemic uncertainty, i.e., the fact that we are not sure if we have selected the correct model parameters. If epistemic uncertainty is large, the models implied by the posterior disagree on a test point and including other models as well makes the confidence decrease. Incorporating epistemic uncertainty by accounting for more than one plausible models (parameter settings) via the BMA of Eq. (5) is the focus of these notes.

1.2.2 Calibration in the regression setting

For regression, the NN outputs a single value with no uncertainty, i.e., it is 100% confident. Calibration in this context relates to taking into account other models as well and thus admitting some uncertainty in the output. A well-calibrated network should predict with high uncertainty far from data, as well as in regions between separated clusters of observations (see Section 2.4). For example, in Fig. 2, the probabilistic model is more calibrated than the standard NN: for a test point (shown with red) far from the training data (left of the dashed blue line) the standard NN confidently predicts a value, whereas the probabilistic model predicts a similar value but with high uncertainty. This test point far from the training data is called out-of-distribution datapoint.

Finally, even if epistemic uncertainty is taken into account by using a BNN, the obtained confidence *may still be miscalibrated.* BNN uncertainty quality as well as recalibration methods are discussed in Section 2.4.

1.2.3 Accuracy

Accuracy refers to the correctness of the NN output. According to Wilson and Izmailov (2020), the MAP approximation of Eq. (11) (as opposed to using the exact Eq. (5)) is reasonable only if $p(\boldsymbol{w}|\mathcal{D},\mathcal{H})$ is peaked around $\hat{\boldsymbol{w}}$ and also $p(y^*|x^*,\boldsymbol{w},\mathcal{H})$ does not vary much where the mass of $p(\boldsymbol{w}|\mathcal{D},\mathcal{H})$ is. However, modern NNs are usually highly underspecified by the available data, and therefore have diffuse likelihoods, not strongly favoring any one setting of parameters. Further, different settings of parameters correspond to a diverse variety of compelling hypotheses for the data (Garipov et al., 2018).

Therefore, this is a setting that can benefit from using a BMA. According to Wilson and Izmailov (2020), even a Gaussian approximation to the posterior (Laplace approximation; see Section 2.2.4) is preferable to the MAP approximation.

2 Components of Bayesian deep learning

As discussed in Blei (2014), building and computing with Bayesian models is an iterative process. This process involves building a model (architecture and priors in BNNs), inferring model parameters (posterior distribution), and criticizing the model (optimizing hyperparameters, comparing different models, re-building the model). This process is briefly discussed in this section.

2.1 Model building

2.1.1 Stochastic NNs and BNNs

As described in Jospin et al. (2020), stochastic NNs are built by introducing stochastic components into a NN. The stochastic components can either be stochastic activations where only a set of weights along with a probability distribution for the activation is learned (Fig. 3-left), or stochastic weights where a probability distribution over the weights is learned (Fig. 3-right). A BNN is a stochastic NN trained using Bayesian inference. In these notes, BNNs with stochastic weights are considered.

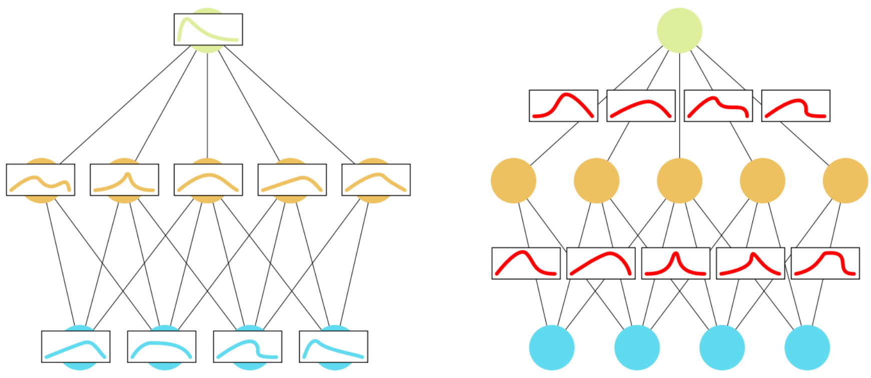


Fig. 3. Figure 3 from Jospin et al. (2020): The stochastic components can either be stochastic activations where only a set of weights along with a probability distribution for the activation is learned (left) or stochastic weights where a probability distribution over the weights is learned (right).

A BNN can be easily understood through its generative process; i.e., the way the stochastic components interact with the functional form of the NN for producing observations. Specifically, for a regression BNN, the data generating process involves drawing a parameter vector from the prior and then for each datapoint drawing an observation from a Gaussian distribution with mean the output of the NN (with the drawn parameter vector) and a pre-specified variance. Inference refers, thus, to the opposite process: having collected a set of datapoints we seek to infer what the drawn parameter vector could have been.

As noted by Wilson and Izmailov (2020), for building a model with satisfactory generalization properties what matters is the distribution over functions (functional prior)

$$p(f_{\mathcal{H}}(x)) = \int f_{\mathcal{H}}(x; \boldsymbol{w}) p(\boldsymbol{w}|\mathcal{H}) d\boldsymbol{w} \tag{25}$$

that is induced by both the functional form of the model $f_{\mathcal{H}}(x; \boldsymbol{w})$ and the parameter prior $p(\boldsymbol{w}|\mathcal{H})$. In this regard, $p(f_{\mathcal{H}}(x))$ depends on the selected network architecture, the noise model, the priors, the grouping of parameters with different priors for each group (see Neal, 1995 p. 77) and the values of the hyperparameters. In Sections 2.1.2-2.1.4, how the parameter prior with the functional form interact is briefly demonstrated.

2.1.2 Gaussian process limit

It has been shown in Neal (1995) that under certain conditions, placing a Gaussian prior on $\boldsymbol{w}$ and taking the number of hidden units of a single-layer NN to infinity results in the NN becoming a Gaussian process (GP). This is based on the central limit theorem (CLT). In this regard, the NN induces some covariance function that can be used with the convenient theory of GPs for exact Bayesian inference. Note that besides its importance, the discussion in Neal (1995) is limited to only proving the above result. Neal (1995) did not eventually obtain the induced covariance function nor did he use this limit in his calculations; he instead used large but finite NNs and obtained his results with approximate inference (see Section 2.2.3.5).

Historical note: Williams (1997) computes the covariance of the infinite network; he actually obtains a neural-net-induced GP (NNGP) for the error and the Gaussian activation functions and performs exact Bayesian inference. Lee et al. (2017) computes the covariance function recursively for other activation functions and coins the term NNGP. Lee et al. (2017) also extends the result of Neal (1995) to deep NNs. Pang et al. (2019) compares NNGPs with GPs and finite NNs, and notes that NNGPs can be more flexible than NN and do not suffer from overfitting because they are Bayesian. Nevertheless, they can address only linear or linearized problems in the context of physics-informed NNs.

However, the aforementioned infinite limit does not imply that GPs can replace NNs (see discussion in MacKay, 1998 p. 30). As Neal (1995) puts it, in the infinite limit the contributions of individual units are all negligible, and consequently, these units do not represent hidden features that capture important aspects of the data. According to Nalisnick (2018), for finite NNs there are "jumps" in the regression line, meaning that the function can change drastically depending on which hidden units are active (see Fig. 4a-left). On the other hand, for infinite width NNs the regression line becomes Brownian motion: it wanders according to very small, independent increments (see Fig. 4a-right). Also, according to Matthews et al. (2018), a GP with a fixed kernel does not use a learned hierarchical representation. Such representations are widely regarded to be essential to the success of deep learning. Instead only a few hyperparameters are learned and it seems unlikely this could offer the same benefits as full representation learning.

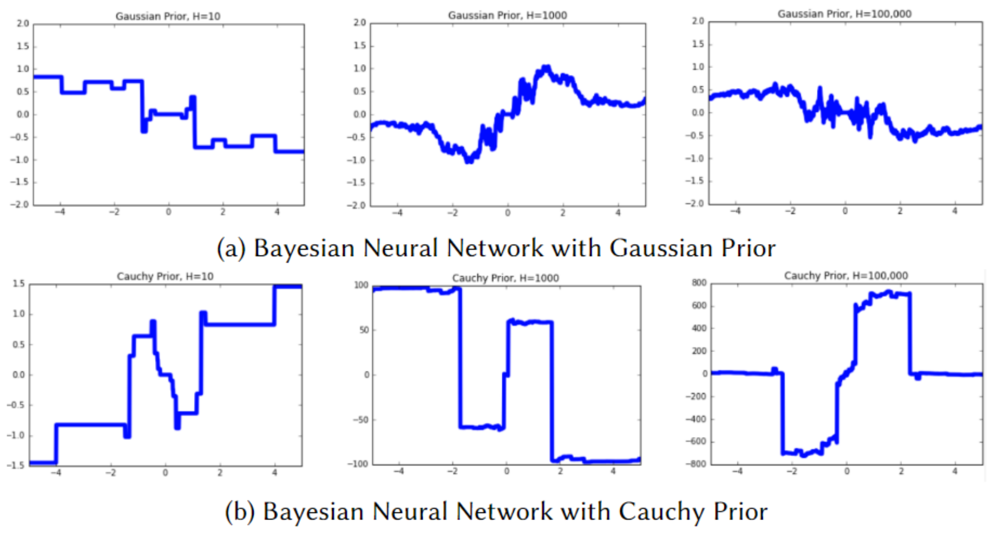


Fig. 4. Figure 3.1 from Nalisnick (2018): GP behavior of BNNs.

The practical implications of the work of Matthews et al. (2018) are worth noting. Specifically, Matthews et al. (2018) has proved the convergence of certain sequences of *finite* fully connected networks with more than one hidden layer to GPs. The authors note that it seems likely that some experiments in the literature studied under the banner of Bayesian deep learning would have given very similar results to a GP with the correct kernel. As a result, they propose that the Bayesian deep learning community routinely compare their results to GPs with the kernels studied in Matthews et al. (2018). According to Nalisnick (2018), this tendency can be a good thing for GP users as it means that BNNs can be used as plug-in approximations for GPs. Swapping a GP for a BNN results in considerable computational savings for large datasets since the user is trading the GP's $\mathcal{O}(N^3)$ cost of matrix inversions and determinants for the BNN's $\mathcal{O}(N)$ training data dependence (see, for example, Flam-Shepherd et al., 2017 where a matching between the GP prior is attempted by using a BNN and minimizing a KL divergence; more information in Section 2.1.4).

In summary, even NNs with discrete weights will converge to GPs via the Lyapunov CLT if the hidden units are real-valued (Matthews et al., 2018). If the full capabilities of BNNs are required, one has to make sure that one or more of the conditions discussed in Matthews et al. (2018) are not satisfied. This can be done by considering CLT-breaking priors discussed in the next section (see Nalisnick, 2018 for an excellent review and discussions).

2.1.3.1 Noninformative/objective priors: These priors aim to inject as little information as possible into the inference procedure. An objective prior is one that has some formal invariance property. The two best known examples of objective priors are Jeffreys' non-informative priors and, their extension, reference priors, which are both invariant to model reparametrization. Reference priors are model-dependent and notoriously difficult to derive for complicated models (Simpson et al., 2015). According to Nalisnick (2018), the mathematical rigor that makes objective priors attractive also makes their use problematic: their derivation is difficult for all but the simplest models. In this regard, Nalisnick (2018) proposes approximate constructions of such priors.

2.1.3.2 Weakly informative priors: Between objective and expert priors lies the realm of "weakly informative" priors. These priors are constructed by recognizing that while you usually do not have strong prior information about the value of a parameter, it is rare to be completely ignorant. This use of weak prior knowledge is often sufficient to regularize the extreme inferences that can be obtained using maximum likelihood (Simpson et al., 2015).

Such priors are typically written in a hierarchical form by assuming they are drawn from a hyperprior (Section 1.1.3). Typically, a Gaussian scale mixture is considered; i.e., the parameters are drawn from a Gaussian with variance drawn from some hyperprior. In this regard, we obtain a Student-t prior by using a Gaussian distribution with scale drawn from an inverse Gamma hyperprior, a Laplace prior by using an exponential hyperprior, and a horseshoe prior by using a half-Cauchy hyperprior (the Cauchy distribution restricted to the positive real numbers).

A Gaussian prior for the parameters is a typical and reasonable choice because the zero-centering induces shrinkage and the Gaussian assumption implies smoothness (Nalisnick, 2018). However, as discussed in Section 2.1.2, even finite-width BNNs may behave like GPs.

According to Nalisnick (2018), one way to break the CLT is to draw the NN weights from a distribution with infinite variance; such as from a member of the symmetric stable family with index less than two. These distributions have heavy, sub-exponential tails, which allows "some of the hidden units in an infinite network [to] have output weights of significant size, allowing them to represent hidden features" (Neal, 1995). Unfortunately, heavy-tailed distributions are not easy to work with analytically or computationally. The Cauchy, for instance, does not have finite moments of any order. Neal (1995) recommends the Student-t distribution as a compromise, which is CLT-breaking when its degrees of freedom parameter is less than two. In Fig. 4b taken from Nalisnick (2018), we see that when a Cauchy prior is used the regression line exhibits *large jumps* even for large widths, which means that the regression function is still dependent on which hidden units are active.

Another choice is the Laplace prior which, due to its dependence on the absolute value of the weights, reflects the symmetry of NNs with respect to the signs of the weights (Williams, 1995). Laplace priors may not provide enough regularization under fully Bayesian treatments of NNs. Note that Laplace priors do not induce the same amount of useful sparsity in their posteriors as is found in their MAP estimates.

Next, the horseshoe prior has no closed-form marginal expression, but it can be easily plotted. This prior allows the NN weights to grow large, like the Student-t, but has fierce shrinkage near zero. These facts make the horseshoe prior a stronger regularizer than the Gaussian, Laplacian, and Student-t densities.

There are also priors without a hierarchical form. For example, a variational-inference-based analysis lead Kingma et al. (2015) to find that the log-uniform distribution is the Bayesian prior that best mimics the dropout mechanism. It is similar to the horseshoe prior but has flatter tails and better sparsity inducing behavior. See Fig. 5 for density comparisons.

2.1.4 Functional priors

As noted in Wilson and Izmailov (2020), although it is popular recently to build directly function-space BNN priors (see $p(f_{\mathcal{H}}(x))$ in Eq. (25)), if we contrive parameter priors $p(\boldsymbol{w})$ to induce distributions over functions $p(f_{\mathcal{H}}(x))$ that resemble familiar models we could be doing the same mistake as when replacing BNNs by GPs. NNs are useful as their own model class precisely because they have different inductive biases from other models. Nevertheless, some techniques pertaining to constructing functional priors are presented in this section.

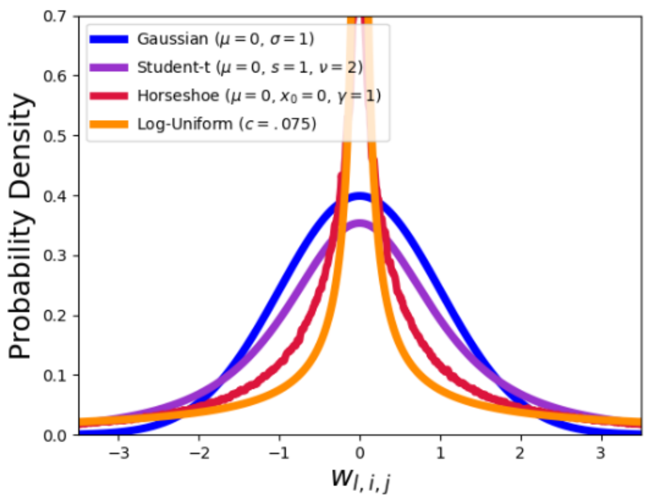


Fig. 5. Figure 3.2a from Nalisnick (2018): Heavy-tailed priors.

Hafner et al. (2018) proposes an implicit functional prior by constructing a data prior. In other words, this prior favors functions that are similar to the data. The authors also perform active learning by selecting new datapoints that maximize the expected information gain. Their prior encourages the BNN to assign high uncertainty (as it should) to datapoints that are out of the distribution that generated the training data (see also Section 2.4). Specifically, a prior $\mathcal{P}$ is constructed according to which each training datapoint is modeled as

$$p(x_i, y_i|\mathcal{P}) = p(x_i|\mathcal{P})p(y_i|x_i, \mathcal{P}) \tag{26}$$

Next, for points $\tilde{x}$ that are perturbations of the training points, the prior gives

$$p(\tilde{x}|\mathcal{P}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{N}(\tilde{x} - x_i|0, \sigma_x^2) \tag{27}$$

and

$$p(\tilde{y}|\tilde{x}, \mathcal{P}) = \mathcal{N}(y, \sigma_y^2) \tag{28}$$

where $\mathcal{N}$ denotes a Gaussian distribution and σ_x^2, σ_y^2 are tunable parameters. Further, y is not clearly defined in the manuscript but its purpose is recovering true labels/values from perturbed inputs $\tilde{x}$. By introducing a data prior, a functional prior is implicitly introduced. To see this, consider the marginal likelihood of a datapoint given as

$$p(y|x) = \int p(y|x, \boldsymbol{w}, \mathcal{H})p(\boldsymbol{w}|\tilde{x}, \tilde{y}, \mathcal{H})p(\tilde{x}, \tilde{y}|\mathcal{P})d\boldsymbol{w}d\tilde{x}d\tilde{y} \tag{29}$$

Clearly, $p(\tilde{x}, \tilde{y}|\mathcal{P})$ induces a parameter prior $p(\boldsymbol{w}|\tilde{x}, \tilde{y}, \mathcal{H})$ which in turn induces a functional prior $p(y|x)$. Finally, training is performed by variational inference where instead of penalizing the posterior for being far from the parameter prior, the posterior predictive

$$q(f_{\mathcal{H}}(\tilde{x})) = \int f_{\mathcal{H}}(\tilde{x}; \boldsymbol{w}) q_{\boldsymbol{\theta}}(\boldsymbol{w}) d\boldsymbol{w} \tag{30}$$

defined using the approximate parameter posterior $q_{\boldsymbol{\theta}}(\boldsymbol{w})$, is penalized for being far from the data prior. Note the similarity of Eq. (30) with Eq. (5) and see also Section 2.2.5 for more information on variational inference.

Next, Flam-Shepherd et al. (2017) trains a BNN to mimic the behavior of a GP prior, by encoding properties of GP priors into BNN weight priors, and performs standard variational inference in the weight space. On the contrary, Sun et al. (2019) have proposed functional variational BNNs for which variational inference is performed directly in the function space and have shown how to also handle functional priors without tractable marginal densities $p(f_{\mathcal{H}})$; i.e., functional priors other than GPs and Student-t processes. For example, they consider piecewise constant functions and piecewise linear functions. Further, Ma et al. (2019) have introduced variational implicit processes which, in a sense, are the reverse of functional variational BNNs. A GP is trained to mimic the behavior of the BNN functional prior and the resulting GP is used for exact Bayesian inference.

Functional neural processes, introduced in Louizos et al. (2019), posit distributions over functions by embedding the dataset into a latent space and constructing a dependency graph among the latent embeddings; recall that in GPs a correlation structure encoded in the covariance matrix measures the similarity between inputs. Finally, Yang et al. (2019) proposes a way of enforcing expert knowledge into the BNN prior in the form of constraints restricting where the function should or should not be.

2.2 Approximate inference

Obtaining the posterior exactly via Eqs. (16) or (17) is computationally and analytically intractable. In this regard, approximate inference refers to approximating the posterior by another distribution and/or obtaining samples from the posterior. As shown in Section 1.1.2, the MAP estimate of Eq. (8) (standard training of NNs) can be seen as an approximate inference technique. However, being a point estimate, the MAP estimate is not Bayesian. In this section, the most widely used Bayesian (or not) approximate inference techniques are briefly reviewed.

2.2.1 Monte Carlo estimation

According to the law of large numbers, for i.i.d. samples $\hat{w}_1, \ldots, \hat{w}_M$ with mean μ the condition

$$\lim_{M \to \infty} P\left(\left| \frac{\sum_{j=1}^{M} \hat{w}_j}{M} - \mu \right| \geq b \right) = 0 \tag{31}$$

is satisfied for any positive b. Similarly,

$$\lim_{M \to \infty} P\left(\left| \frac{\sum_{j=1}^{M} g(\hat{w}_j)}{M} - \mathbb{E}_w[g(w)] \right| \geq b \right) = 0 \tag{32}$$

As a result, the predictive distribution of Eq. (1) can be approximated by a Monte Carlo (MC) estimate as

$$p(y^*|x^*, \mathcal{D}, \mathcal{H}) = \mathbb{E}_{\boldsymbol{w}|\mathcal{D}}[p(y^*|x^*, \boldsymbol{w}, \mathcal{D}, \mathcal{H})] \approx \frac{1}{M} \sum_{j=1}^{M} p(y^*|x^*, \hat{\boldsymbol{w}}_j, \mathcal{D}, \mathcal{H}) \tag{33}$$

where $\{\hat{\boldsymbol{w}}_j\}_{j=1}^{M}$ are samples from the posterior $p(\boldsymbol{w}|\mathcal{D}, \mathcal{H})$.

If also hyperparameters are included as in Eq. (19), $p(y^*|x^*, \mathcal{D}, \mathcal{H})$ can be approximated as

$$p(y^*|x^*, \mathcal{D}, \mathcal{H}) = \mathbb{E}_{\boldsymbol{w},\alpha,\beta|\mathcal{D}}[p(y^*|x^*, \boldsymbol{w}, \mathcal{D}, \beta, \mathcal{H})] \approx \frac{1}{M} \sum_{j=1}^{M} p(y^*|x^*, \hat{\boldsymbol{w}}_j, \hat{\beta}_j, \mathcal{D}, \mathcal{H}) \tag{34}$$

where $\{\hat{\boldsymbol{w}}_j, \hat{\beta}_j\}_{j=1}^{M}$ are samples from the posterior $p(\boldsymbol{w}, \alpha, \beta|\mathcal{D}, \mathcal{H})$ (α is used to draw values for $\boldsymbol{w}$). If, instead, the hyperparameters are optimized (see Section 2.3.2), then $p(y^*|x^*, \mathcal{D}, \mathcal{H})$ can be approximated as

$$p(y^*|x^*, \mathcal{D}, \mathcal{H}) = \mathbb{E}_{\boldsymbol{w}|\mathcal{D},\hat{\alpha},\hat{\beta}}[p(y^*|x^*, \boldsymbol{w}, \mathcal{D}, \hat{\beta}, \mathcal{H})] \approx \frac{1}{M} \sum_{j=1}^{M} p(y^*|x^*, \hat{\boldsymbol{w}}_j, \hat{\beta}, \mathcal{D}, \mathcal{H}) \tag{35}$$

where $\{\hat{\boldsymbol{w}}_j\}_{j=1}^{M}$ are samples from the posterior $p(\boldsymbol{w}|\mathcal{D}, \hat{\alpha}, \mathcal{H})$ and $\hat{\alpha}$, $\hat{\beta}$ are the optimal hyperparameters.

Therefore, using one of the above MC estimates transforms the original problem into a sampling problem: we need to draw independent samples $\{\hat{\boldsymbol{w}}_j\}_{j=1}^{M}$ from the posterior. Drawing independent samples is not straightforward. Fortunately, the above MC estimates (typically with slower convergence) still hold for moderately dependent samples from the posterior (see Neal, 1995 p. 27).

Finally, note that, instead of Eq. (1), an approximation of Eq. (3) or (5) may be required. Thus, to make the formulation more general, the more diverse problem of approximating

$$\mathbb{E}_{\boldsymbol{w}|\mathcal{D}}[g(\boldsymbol{w})] = \int g(\boldsymbol{w})p(\boldsymbol{w}|\mathcal{D},\mathcal{H})d\boldsymbol{w} \tag{36}$$

is considered herein. Typically, the posterior is written as

$$p(\boldsymbol{w}|\mathcal{D},\mathcal{H}) = \frac{f(\boldsymbol{w})}{Z} = \frac{\exp(-\mathcal{L}(\boldsymbol{w}))}{Z} \tag{37}$$

where $f(\boldsymbol{w}) = p(\mathcal{D}|\boldsymbol{w},\mathcal{H})p(\boldsymbol{w}|\mathcal{H})$ and Z is a normalization constant. Therefore, Eq. (36) can also be expressed as

$$\mathbb{E}_{\boldsymbol{w}|\mathcal{D}}[g(\boldsymbol{w})] = \frac{\int g(\boldsymbol{w})f(\boldsymbol{w})d\boldsymbol{w}}{\int f(\boldsymbol{w})d\boldsymbol{w}} \tag{38}$$

MC methods have been widely used in statistical physics. For this reason, the terminology originating from their use in statistical physics often accompanies them even when used in domains other than physics. For example, the distribution of Eq. (37) is called canonical distribution over microstates $\boldsymbol{w}$, $\mathcal{L}(\boldsymbol{w})$ is the energy of the system defined for each microstate $\boldsymbol{w}$, and Z, the normalization constant, is known as the partition function. See Neal (1993), Section 2.4, for more information on this correspondence.

2.2.2 Rejection and importance sampling

This section follows closely the review paper Neal (1993). Suppose that there is a function $q(\boldsymbol{w})$, that is proportional to a density easy to sample from, and that there is also a constant c such that $f(\boldsymbol{w}) < cq(\boldsymbol{w})$ is satisfied for any $\boldsymbol{w}$. Then, for obtaining samples from $f(\boldsymbol{w})$ we can generate a candidate sample $\boldsymbol{w}^*$ from $q(\boldsymbol{w})$ and accept it with probability $p = f(\boldsymbol{w}^*)/cq(\boldsymbol{w}^*)$. This is known as rejection sampling and is equivalent to drawing also some u from a uniform distribution in $[0, 1)$ and accepting $\boldsymbol{w}^*$ if $u < p$. Unfortunately, it is almost impossible to find an appropriate $q(\boldsymbol{w})$ with small c.

Suppose, next, that there is a density $q(\boldsymbol{w})$ (not necessarily normalized) that is nonzero where $f(\boldsymbol{w})$ is nonzero. Then, following Eq. (38), $\mathbb{E}_{\boldsymbol{w}|\mathcal{D}}[g(\boldsymbol{w})]$ can be written as

$$\mathbb{E}_{\boldsymbol{w}|\mathcal{D}}[g(\boldsymbol{w})] = \frac{\int g(\boldsymbol{w})\frac{f(\boldsymbol{w})}{q(\boldsymbol{w})}q(\boldsymbol{w})d\boldsymbol{w}}{\int \frac{f(\boldsymbol{w})}{q(\boldsymbol{w})}q(\boldsymbol{w})d\boldsymbol{w}} = \frac{\mathbb{E}_{\boldsymbol{w}\sim q}[g(\boldsymbol{w})\frac{f(\boldsymbol{w})}{q(\boldsymbol{w})}]}{\mathbb{E}_{\boldsymbol{w}\sim q}[\frac{f(\boldsymbol{w})}{q(\boldsymbol{w})}]} \tag{39}$$

which can be approximated in a similar way as Eq. (33) (see Eq. (3.6) in Neal, 1993). If $q(\boldsymbol{w})$ is a good approximation of $f(\boldsymbol{w})$ then an MC estimate of Eq. (39) will be a good estimate of $\mathbb{E}_{\boldsymbol{w}|\mathcal{D}}[g(\boldsymbol{w})]$. However, guessing such a density $q(\boldsymbol{w})$ is not easy and thus, this technique, known as simple importance sampling, is not applicable in the context of deep learning.

2.2.3 Markov chain sampling

This section follows closely the review paper Neal (1993) and the more compact summary in Neal (1995).

2.2.3.1 Markov chains: A sequence of dependent $\boldsymbol{w}$ values can be generated using a Markov chain with $p(\boldsymbol{w}|\mathcal{D},\mathcal{H})$ as its invariant (stationary) distribution; this idea defines a family of techniques known as Markov chain Monte Carlo (MCMC). The first state of the chain is $\boldsymbol{w}^{(1)}$ and it has some initial distribution. A transition probability is also defined for each state t denoted by $T_t(\boldsymbol{w}^{(t+1)}|\boldsymbol{w}^{(t)})$. If T_t does not depend on t then the Markov chain is called homogeneous with transition probability T. An invariant or stationary distribution q is one for which if $\boldsymbol{w}^{(t)}$ has distribution q, then any $\boldsymbol{w}^{(t')}$ with $t' > t$ will also have distribution q. An ergodic Markov chain has a unique invariant distribution called equilibrium distribution. Whatever the initial state $\boldsymbol{w}^{(1)}$ is, the same equilibrium distribution will be reached in stationarity.

Therefore, if we can construct an ergodic Markov chain with equilibrium distribution $q(\boldsymbol{w}) = p(\boldsymbol{w}|\mathcal{D},\mathcal{H})$, then we can sample from $p(\boldsymbol{w}|\mathcal{D},\mathcal{H})$ by sampling from $q(\boldsymbol{w})$; this can be performed by simulating the Markov chain and discarding some "burn-in" states until it reaches equilibrium. After we sample one state then we can start over the Markov chain with the sampled state as the initial state. Due to ergodicity, whatever the sampled state was, after a few steps the equilibrium distribution q will be reached again. We can thus *resample from the same realization* after some "lag" steps.

One way to construct an ergodic Markov chain is by applying a set of base transitions $B_1, \ldots, B_k$ in turn (e.g., by changing only a subset of the dimensions of $\boldsymbol{w}^{(t)}$ in each transition). In this regard, q needs to remain invariant with respect to each transition, so that it remains invariant with respect to the overall transition $T = B_1 \cdots B_k$, where a center dot denotes sequential transitions. This is utilized, for instance, in Gibbs sampling.

As an illustrative example, consider the homogeneous Markov chain that starts from state 0 and moves left or right ± 1 with probability $1/4$ and stays at the same state with probability $1/2$. Suppose also that it is confined in the finite space $\{-5, \ldots, 5\}$. This confined random walk is ergodic with the uniform distribution as its equilibrium distribution. As a result, a sample from the uniform distribution can be obtained by

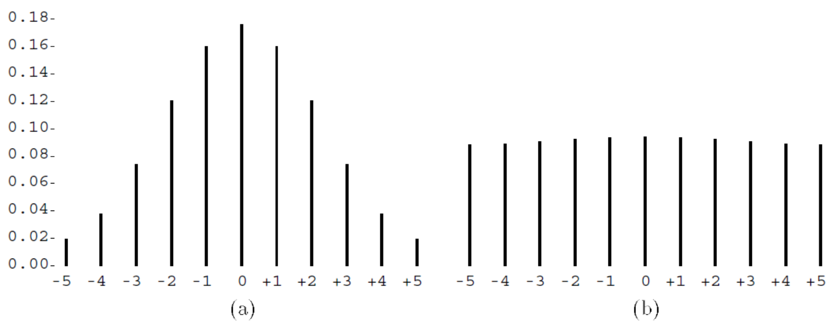


Fig. 6. Figure 3.2 from Neal (1993): Distribution of confined random walk states (a) after 10 iterations and (b) after 100 iterations. A sample from the uniform distribution can be obtained by simulating the random walk for some sufficiently large number of iterations.

simulating the random walk for some sufficiently large number of iterations. This is depicted in Fig. 6 for two different numbers of iterations.

2.2.3.2 Gibbs sampling: Possibly the simplest Markov chain-based sampling scheme is Gibbs sampling. For a k-dimensional $\boldsymbol{w}$ vector, k transitions $B_1, \ldots, B_k$ are considered; in each one only one dimension of the t-th state $\boldsymbol{w}^{(t)}$ is changed using the conditional distribution (under $p(\boldsymbol{w}|\mathcal{D}, \mathcal{H})$) of each dimension given the current values of all the rest. This technique is widely used in the context of latent variable models (e.g., Blei, 2014), but in deep learning the conditional distributions under the posterior are typically not available.

2.2.3.3 Metropolis algorithm: A new state $\boldsymbol{w}^{(t+1)}$ is generated from $\boldsymbol{w}^{(t)}$ by first generating a candidate state using a proposal distribution $S(\boldsymbol{w}^*|\boldsymbol{w}^{(t)})$ and then deciding whether to accept the candidate or not. If it is not accepted, $\boldsymbol{w}^{(t+1)}$ is taken to be $\boldsymbol{w}^{(t)}$. The steps, thus, are

1. generate $\boldsymbol{w}^*$ from some proposal distribution $S(\boldsymbol{w}^*|\boldsymbol{w}^{(t)})$
2. if $\frac{p(\boldsymbol{w}^*|\mathcal{D},\mathcal{H})}{p(\boldsymbol{w}^{(t)}|\mathcal{D},\mathcal{H})} = \exp\big(-\mathcal{L}(\boldsymbol{w}^*) + \mathcal{L}(\boldsymbol{w}^{(t)})\big) \geq 1$, then accept $\boldsymbol{w}^*$, otherwise accept $\boldsymbol{w}^*$ with probability $p = \exp\big(-\mathcal{L}(\boldsymbol{w}^*) + \mathcal{L}(\boldsymbol{w}^{(t)})\big)$
3. if $\boldsymbol{w}^*$ is accepted, then set $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^*$, otherwise set $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)}$.

Although the form of the acceptance probability can be different than the one in Step 2, acceptance or rejection of the candidate state is typically based on the change of "energy" $\mathcal{L}$. The proposal distribution $S(\boldsymbol{w}^*|\boldsymbol{w}^{(t)})$, among many choices, can be a Gaussian distribution centered at $\boldsymbol{w}^{(t)}$.

2.2.3.4 The stochastic dynamics method: The canonical distribution for the "position" variable $\boldsymbol{w}$ of the corresponding to statistical inference physical system is defined in Eq. (37). For formulating the stochastic dynamics method, a "momentum" variable $\boldsymbol{p}$ is also introduced for which there is one-to-one correspondence with the components of $\boldsymbol{w}$. The canonical distribution for the phase space of $\boldsymbol{w}$ and $\boldsymbol{p}$ is defined as

$$p(\boldsymbol{w}, \boldsymbol{p}) \propto \exp(-H(\boldsymbol{w}, \boldsymbol{p})) \tag{40}$$

where $H(\boldsymbol{w}, \boldsymbol{p}) = \mathcal{L}(\boldsymbol{w}) + K(\boldsymbol{p})$, with K denoting the "kinetic" energy, is the total energy of the system and is known as the Hamiltonian function. In this regard, $\boldsymbol{w}$ and $\boldsymbol{p}$ are independent, i.e., the marginal of $\boldsymbol{w}$ under $p(\boldsymbol{w}, \boldsymbol{p})$ is the same as the one we seek to sample from. Thus, a Markov chain that converges to $p(\boldsymbol{w}, \boldsymbol{p})$ can be constructed and $\boldsymbol{w}$ samples can be collected by sampling from $p(\boldsymbol{w}, \boldsymbol{p})$ and discarding the $\boldsymbol{p}$ samples.

In the stochastic dynamics method there are two sub-tasks: 1) sampling states $(\boldsymbol{w}, \boldsymbol{p})$ with fixed energy H, and 2) sampling states with different values of H. Loosely speaking, in the first task we obtain different samples of $\boldsymbol{w}$ by simply following the Hamiltonian dynamics of the system without visiting states of different probability, and in the second task we visit states with different probability by changing only the state of $\boldsymbol{p}$, which is straightforward. Task 1 is done by simulating the Hamiltonian dynamics of the system, in which the state $(\boldsymbol{w}, \boldsymbol{p})$ evolves in fictitious time τ. It can be shown that H stays the same and thus, $p(\boldsymbol{w}, \boldsymbol{p})$ is invariant with respect to transitions related to following a trajectory based on the Hamiltonian dynamics equations. In many cases these transitions will eventually explore the whole range of phase space with fixed H. Task 2 is done by Gibbs sampling updates of the momentum via its density which is proportional to $\exp(-K(\boldsymbol{p}))$. If, for example, $K(\boldsymbol{p})$ is taken to be $K(\boldsymbol{p}) = \|\boldsymbol{p}\|_2^2/2$, then the conditional distributions required by Gibbs sampling are readily available (the components of $\boldsymbol{p}$ have independent Gaussian distributions).

In practice, Hamiltonian dynamics are simulated by some discretization using finite time steps (e.g., leapfrog method). To follow the dynamics for some period of fictitious time $\Delta\tau$ (see Neal, 1993 for selecting $\Delta\tau$), $T = \Delta\tau/\epsilon$ iterations are applied, with ϵ the time step. Using the leapfrog method, H does not stay exactly the same; MC estimates obtained via the stochastic dynamics method will have some systematic error that goes to zero as ϵ goes to zero. This systematic error is eliminated in the hybrid MC (HMC; also known as Hamiltonian MC) algorithm by merging the stochastic dynamics method with the Metropolis algorithm.

2.2.3.5 Hybrid Monte Carlo: The dynamical transitions in the HMC algorithm are performed with two modifications: 1) a random decision is made for each transition whether to simulate the dynamics forward or backward in time, and 2) the state reached is only a candidate; it is accepted based on the change in total energy (as in the Metropolis algorithm). The rejections eliminate the bias introduced by inexact simulation. The values of ϵ and T may be chosen at random based on a fixed distribution. HMC is often used as the "ground truth" for comparing different approximate inference techniques; see also Betancourt (2017).

2.2.3.6 Stochastic gradient Langevin dynamics: The Langevin MC method is an HMC method with $T = 1$; i.e., with a single leapfrog iteration; see, however, Neal (1993) for the benefits of using larger values of T. In fact, in the uncorrected version of Langevin MC (all candidates accepted) there is no need of the two-step approach described above and no need for explicitly representing the momentum at all; one stochastic differential equation (Langevin equation; see Eq. (5.28) in Neal, 1993) is used for picking new values of $\boldsymbol{w}$.

Note that all the MCMC methods described so far require computations over the whole dataset at every iteration, resulting in high computational cost for large datasets. In this regard, motivated by the stochastic gradient descent (SGD) algorithm, in which at each iteration only a subset (mini-batch) of the available data is used, Welling and Teh (2011) proposed a technique that combines Langevin MC and SGD. Specifically, the resulting weight update is computed using only a subset of the data and is the same as the update in SGD but with added Gaussian noise.

Further, also motivated by the scalability issues of MCMC techniques, Chen et al. (2014) introduced stochastic gradients within the HMC framework. An investigation of MCMC techniques that rely on stochastic gradients can be found in Mandt et al. (2017). Finally, see Section 2.2.5.3 for similar adaptations of sub-sampling within the context of variational inference.

2.2.4 Laplace approximation

The Laplace method for approximating the posterior of a BNN was proposed by Denker and LeCun (1991). In short, for fixed hyperparameters, they optimized the NN weights as in Eq. (9) and fitted a Gaussian distribution to the discovered mode, with width determined by the Hessian of the training error. This is equivalent to using a quadratic approximation for the log of the probability density. In this regard, the covariance matrix of the fitted distribution is the inverse of the Hessian $\boldsymbol{A}$ of the loss, given as $\boldsymbol{A} = -\nabla\nabla \log p(\hat{\boldsymbol{w}}|\mathcal{D}, \alpha, \beta, \mathcal{H})$, and the posterior $p(\boldsymbol{w}|\mathcal{D}, \alpha, \beta, \mathcal{H})$ for given α, β, is approximated as (see Eq. (19) in MacKay, 1995b))

$$q(\boldsymbol{w}) = \frac{1}{Z} \exp\left(-\mathcal{L}(\hat{\boldsymbol{w}}) - \frac{1}{2}(\boldsymbol{w} - \hat{\boldsymbol{w}})^T \boldsymbol{A}(\boldsymbol{w} - \hat{\boldsymbol{w}}) \right) \tag{41}$$

where Z is a normalization constant. Because of the analytical convenience offered by Gaussian distributions, selecting hyperparameters and determining the predictive distribution of Eq. (1) (or its simplifications Eqs. (3) and (5)) is straightforward. For example, the integral of Eq. (1) can be determined analytically by linearizing locally the output (e.g., Section 5.2 in MacKay, 1995b). Alternatively, an MC estimate of the predictive distribution can be obtained by drawing M samples from the approximate Gaussian posterior $q(\boldsymbol{w})$, i.e., $\{\hat{\boldsymbol{w}}_j\}_{j=1}^{M}$, and summing over the M different predictions as in Eq. (33). Finally, one can also obtain unbiased estimates of the form of Eq. (33) via importance sampling using the approximating Gaussian density as the sampling distribution (see Eq. (39) herein and Eq. (3.6) in Neal, 1993).

In terms of computational efficiency, Ritter et al. (2018) show how the Laplace approximation can be performed efficiently by using Kronecker factored covariance matrices. In terms of accuracy, Neal (1995) criticized the simplistic unimodal approximation of the posterior. Ritter et al. (2018) compare their version of Laplace approximation with dropout and show that the results can be at least as good. Finally, in Section 4.1 in MacKay (1995b) modifications are proposed for treating multimodal posteriors; in short, a different Gaussian distribution with its own hyperparameters is fitted for every mode of the posterior. This multimodal approximation seems similar to deep ensembles of Lakshminarayanan et al. (2017) (Section 2.2.6) and to multi-SWAG of Wilson and Izmailov (2020) (Section 2.2.7). Multimodal Laplace approximation seems too computationally costly in modern deep learning applications, but more literature review is required to verify this. See Ashukha et al. (2020) for a comparison of approximate inference techniques which, among others, includes Laplace approximation and deep ensembles.

2.2.5 Variational inference

In variational inference (VI), also known as ensemble learning, the posterior is approximated by a so-called variational distribution $q_{\boldsymbol{\theta}}(\boldsymbol{w})$ parametrized by $\boldsymbol{\theta}$. In general, we start with the objective to maximize the probability of observing the training dataset $\mathcal{D}$ under the model $\mathcal{H}$; i.e., to maximize $p(\mathcal{D}|\alpha, \beta, \mathcal{H})$ of Eq. (17) for given hyperparameters α, β (omitted for the rest of this section). In this regard, noting that $\log p(\mathcal{D}|\mathcal{H})$ is constant with respect to $q_{\boldsymbol{\theta}}(\boldsymbol{w})$, $\log p(\mathcal{D}|\mathcal{H})$ is written as

$$\log p(\mathcal{D}|\mathcal{H}) = \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}\left[\log p(\mathcal{D}|\mathcal{H})\right] \tag{42}$$

or using $p(\mathcal{D}, \boldsymbol{w}|\mathcal{H}) = p(\mathcal{D}|\mathcal{H})p(\boldsymbol{w}|\mathcal{D}, \mathcal{H})$,

$$\log p(\mathcal{D}|\mathcal{H}) = \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}\left[\log\left(\frac{p(\mathcal{D}, \boldsymbol{w}|\mathcal{H})}{p(\boldsymbol{w}|\mathcal{D}, \mathcal{H})}\frac{q_{\boldsymbol{\theta}}(\boldsymbol{w})}{q_{\boldsymbol{\theta}}(\boldsymbol{w})}\right)\right] \tag{43}$$

or

$$\log p(\mathcal{D}|\mathcal{H}) = ELBO(\boldsymbol{\theta}) + KL\left(q_{\boldsymbol{\theta}}(\boldsymbol{w})||p(\boldsymbol{w}|\mathcal{D}, \mathcal{H})\right) \tag{44}$$

where $KL\left(q_{\boldsymbol{\theta}}(\boldsymbol{w})||p(\boldsymbol{w}|\mathcal{D}, \mathcal{H})\right)$ denotes the KL divergence between the true and the approximate posterior, and the evidence lower bound (ELBO) is given as

$$ELBO(\boldsymbol{\theta}) = \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}\left[\log\left(\frac{p(\mathcal{D}, \boldsymbol{w}|\mathcal{H})}{q_{\boldsymbol{\theta}}(\boldsymbol{w})}\right)\right] \leq \log p(\mathcal{D}|\mathcal{H}) \tag{45}$$

By maximizing the ELBO, simultaneously the evidence of the model is maximized and the KL divergence between the true and the approximate posterior is minimized. Note, also, that Eq. (45) can equivalently be written as

$$ELBO(\boldsymbol{\theta}) = \int \log\left(p(\mathcal{D}|\boldsymbol{w}, \mathcal{H})\right) q_{\boldsymbol{\theta}}(\boldsymbol{w})d\boldsymbol{w} - KL\left(q_{\boldsymbol{\theta}}(\boldsymbol{w})||p(\boldsymbol{w}|\mathcal{H})\right) \tag{46}$$

i.e., maximizing the ELBO encourages $q_{\boldsymbol{\theta}}(\boldsymbol{w})$ to explain the data well, while being as close to the prior as possible (penalizing overcomplex models). Note, finally, that the quality of the approximation depends on the expressiveness of $q_{\boldsymbol{\theta}}(\boldsymbol{w})$. As noted in Barber and Bishop (1998), the richer the family of q distributions considered, the better the resulting bound will be (see Foong et al., 2019a for theoretical results). As a side note, see Mescheder et al. (2018) for a GAN-based technique for obtaining an expressive posterior.

Having determined the optimal parameters $\boldsymbol{\theta}$, $p(\boldsymbol{w}|\mathcal{D},\mathcal{H})$ in Eq. (1) is substituted by $q_{\boldsymbol{\theta}}(\boldsymbol{w})$, which is easy to sample from. An MC estimate of the predictive distribution can be obtained via Eq. (33) by drawing M samples from the approximate posterior $q_{\boldsymbol{\theta}}(\boldsymbol{w})$.

2.2.5.1 Computational efficiency and tractability: Eq. (46) can also be written as

$$\mathcal{L}_{VI}(\boldsymbol{\theta}) = -\sum_{i=1}^{N} \int \log\left(p(y_i|x_i, \boldsymbol{w}, \mathcal{H})\right) q_{\boldsymbol{\theta}}(\boldsymbol{w}) d\boldsymbol{w} + KL\left(q_{\boldsymbol{\theta}}(\boldsymbol{w})||p(\boldsymbol{w}|\mathcal{H})\right) \tag{47}$$

where $\mathcal{L}_{VI}(\boldsymbol{\theta}) = -ELBO(\boldsymbol{\theta})$ is the minimization objective. As reviewed in Gal (2016), evaluating this objective poses two main difficulties. First, the integral terms in the summation are not tractable and second, computations over the entire dataset are costly. For the first, MC estimators are utilized, whereas for the second data sub-sampling is utilized.

2.2.5.2 MC estimators in VI: Note that the MC estimators discussed in this section are distinct from the ones discussed in a different context in Section 2.2.1. We seek an estimator of the integral term in Eq. (47) and, more specifically, for its derivative (gradient) with respect to $\boldsymbol{w}$ in order to optimize it (see Gal, 2016 Section 3.1.1 for a detailed discussion). In one dimension ($k = 1$ parameter), this derivative has the form

$$I(\theta) = \frac{\partial}{\partial \theta} \int f(w) q_\theta(w) dw \tag{48}$$

One estimator arises by writing

$$I(\theta) = \frac{\partial}{\partial \theta} \int f(w) q_\theta(w) dw = \int f(w) \frac{\partial \log q_\theta(w)}{\partial \theta} q_\theta(w) dw \tag{49}$$

and thus, we draw w from $q_\theta(w)$ and estimate the derivative by $f(w)\frac{\partial \log q_\theta(w)}{\partial \theta}$. This is known as score function or likelihood ratio estimator.

Another estimator arises by using the reparametrization trick. Specifically, writing

$$q_\theta(w) = \int q_\theta(w, \epsilon) d\epsilon = \int q_\theta(w|\epsilon) p(\epsilon) d\epsilon \tag{50}$$

where $q_\theta(w|\epsilon) = \delta(w - g(\theta, \epsilon))$ for some transformation $w = g(\theta, \epsilon)$, we get

$$\int f(w) q_\theta(w) dw = \int f(g(\theta, \epsilon)) p(\epsilon) d\epsilon \tag{51}$$

and thus,

$$I(\theta) = \frac{\partial}{\partial \theta} \int f(w) q_\theta(w) dw = \int \frac{d}{dw} f(g(\theta, \epsilon)) \frac{\partial}{\partial \theta} g(\theta, \epsilon) p(\epsilon) d\epsilon \tag{52}$$

Therefore, we draw ϵ from $p(\epsilon)$ and estimate the derivative $I(\theta)$ by $f'(g(\theta,\epsilon))\frac{\partial}{\partial\theta}g(\theta,\epsilon)$; f' denotes the derivative of f with respect to w. This is known as path-wise estimator.

Lastly, if $q_\theta(w)$ is Gaussian, then an alternative estimator, called characteristic function estimator involving the second derivative of f, can be constructed.

See Gal (2016) for an analysis on the variances of the above estimators.

2.2.5.3 Data sub-sampling in VI: If only datapoints in the index set S of size $|S|$ are considered, Eq. (47) is approximated as

$$\mathcal{L}_{VI}(\boldsymbol{\theta}) = -\frac{N}{|S|} \sum_{i \in S} \int \log\left(p(y_i|x_i, \boldsymbol{w}, \mathcal{H})\right) q_{\boldsymbol{\theta}}(\boldsymbol{w}) d\boldsymbol{w} + KL\left(q_{\boldsymbol{\theta}}(\boldsymbol{w}) || p(\boldsymbol{w}|\mathcal{H})\right) \tag{53}$$

2.2.5.4 Historical note: The technique proposed by Hinton and Van Camp (1993) can be seen as the first VI approximation to BNNs; they considered a Gaussian $q_{\boldsymbol{\theta}}(\boldsymbol{w})$ with diagonal covariance matrix (also known as mean-field VI). Barber and Bishop (1998) replaced the diagonal with a full covariance matrix, i.e., modeled correlations between the weights; this resulted to both increased accuracy and computational cost.

In terms of MC estimators in VI (Section 2.2.5.2), Graves (2011) approximated the intractable expected log likelihood with a characteristic function estimator. Blundell et al. (2015) utilized the path-wise estimator; this technique is often referred to as Bayes by backprop (see also Kucukelbir et al., 2017).

In terms of data sub-sampling (Section 2.2.5.3), stochastic VI in the context of deep learning and VI was proposed by Hoffman et al. (2013). Graves (2011) did use data sub-sampling but in a different context.

2.2.5.5 Stochastic regularization techniques (dropout): These techniques are used to regularize NNs through the injection of stochastic noise into the model. One of them, dropout, is trained exactly as standard NNs (including standard weight-decay regularization), but in every iteration some of the parameters are set to zero when computing the gradient descent step. Equivalently, the weight matrices are multiplied by diagonal matrices with 1's and 0's in the diagonal for randomly deleting weights in each iteration.

Gal (2016) has shown that by expressing the above matrix multiplication as reparametrization (see Section 2.2.5.2), dropout is equivalent with BNNs trained with stochastic VI, given that the prior over the weights and the approximate posterior in VI have a specific form (see Section 3.2.3 in Gal, 2016). NNs trained with dropout can thus be used *exactly as BNNs*; they can be used for making predictions using Eq. (33) (multiple forward passes), where $\{\hat{\boldsymbol{w}}_j\}_{j=1}^{M}$ are obtained by multiplying the optimal weights by binary vectors drawn from a Bernoulli distribution. In other words, a set of optimal weights are first obtained by training using dropout and then samples from the approximate posterior are just noisy versions of these weights. This is called MC-dropout in order to be distinguished from "standard" dropout. Standard dropout approximates the model average of Eq. (33) using only one forward pass with the obtained sets of weights multiplied by a deterministic factor. Finally, different reparametrizations lead to different versions of dropout, e.g., Gaussian dropout.

2.2.6 Deep ensembles

Approximate inference by deep ensembles pertains to obtaining M different settings of weights $\boldsymbol{w}$ by training independently a NN M times. Although it is considered a non-Bayesian technique, Wilson and Izmailov (2020) states that deep ensembles are closely related to the Bayesian framework: one hypothesis (parameter setting) is considered correct, but we do not have enough data to fully specify the parameters. Because deep ensembles and the Bayesian framework have similar mindsets, deep ensembles can be seen as approximating the BMA of Eq. (1). Precisely, deep ensembles are construed as approximating the posterior $p(\boldsymbol{w}|\mathcal{D}, \alpha, \beta, \mathcal{H})$ by a mixture of delta distributions written as

$$q(\boldsymbol{w}) = \frac{1}{M}\sum_{j=1}^{M}\delta(\boldsymbol{w} - \hat{\boldsymbol{w}}_j) \tag{54}$$

where $\{\hat{\boldsymbol{w}}_j\}_{j=1}^{M}$ are different MAP estimates arising from random initialization of the NN. Therefore, Eq. (1) is approximated by Eq. (33). Note that the M values of $\hat{\boldsymbol{w}}$ in this context are not samples from an approximate posterior, but rather M modes of the true posterior obtained by M SGD runs.

2.2.7 SGD-based techniques

Snapshot ensembles (Huang et al., 2017) is a simple example of an array of methods which collect samples from a single SGD training trajectory of a network in weight space to construct an ensemble (*no additional cost* as compared to standard NN training). The main idea is to use a cyclical learning rate and perform sampling of the weights at the end of each cycle; i.e., the learning rate is reduced up to a certain point, a sample is collected, and then the learning rate jumps back to a large value (so that a very different weight is visited next) before it drops again.

Garipov et al. (2018) proposes a technique for finding, in the weight space, paths of low training loss and test error. Using these paths they show that moving away from a weight obtained via SGD even by a small distance along the path produces diverse predictions with similar test error. With this in mind, they propose a cyclical learning rate with higher frequency than the one used in Huang et al. (2017) (i.e., they explore a smaller region of the weight space) and sample different weights at the end of each cycle. This technique is called fast geometric ensembling. Both of the aforementioned techniques use the sampled weights in an *ensemble of models*; i.e., they average the predictions of the obtained models via Eq. (33). This requires multiple forward passes for making a prediction.

An alternative approach is to take the average of the sampled weights and use it for making predictions (Izmailov et al., 2019). This approach, known as stochastic weight averaging (SWA), can be seen as an approximation of ensembling and it is more computationally efficient (one forward pass required for each test case). Next, Maddox et al. (2019) proposes a modification of SWA, known as SWA-Gaussian (SWAG), in which a Gaussian distribution is fitted on the collected weight samples (using the moments of the samples, not as in Laplace approximation). They use this distribution as an approximate posterior to sample from and make predictions via Eq. (33). Finally, Wilson and Izmailov (2020) proposes an approach that combines deep ensembles of Section 2.2.6 and SWAG, called multi-SWAG; a Gaussian distribution is fitted for every SGD trajectory of deep ensembles, leading thus to a mixture of Gaussian distributions.

Note that snapshot ensembles, fast geometric ensembling, SWA and SWAG do not introduce any additional computational burden to the training of NNs. Similarly,

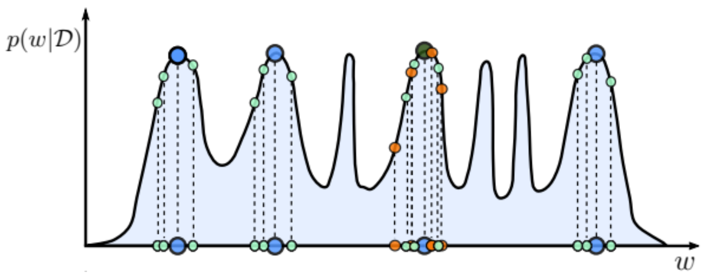


Fig. 7. Figure 3 from Wilson and Izmailov (2020): How deep ensembles (blue), VI (orange), and multi-SWAG (green) sample from the posterior. Deep ensembles only find different modes, VI finds one mode and explores the nearby region, and multi-SWAG finds different modes and explores the nearby regions.

multi-SWAG does not introduce any additional cost as compared to deep ensembles.

See Fig. 7 for an illustrative comparison of deep ensembles, VI, and multi-SWAG for posterior sampling.

2.3 Model comparison

Model comparison refers to selecting between different priors and hyperpriors, between different hyperparameter values and between different models $\mathcal{H}$.

2.3.1 Hyperparameters: optimize, sample or integrate out upfront?

As shown in Section 1.1.3, the posterior $p(\boldsymbol{w}|\mathcal{D},\mathcal{H})$ is obtained by integrating out the hyperparameters based on their posterior $p(\alpha,\beta|\mathcal{D},\mathcal{H})$ given by Eq. (18). According to this procedure, there are two levels of inference, i.e., inference of the hyperparameters via Eq. (18) and inference of the parameters via Eq. (17) given the inferred hyperparameters. However, the integral of Eq. (16) is typically intractable.

One workaround is to approximate Eq. (16) as

$$p(\boldsymbol{w}|\mathcal{D},\mathcal{H}) \approx p(\boldsymbol{w}|\mathcal{D},\hat{\alpha},\hat{\beta},\mathcal{H}) \tag{55}$$

where $\hat{\alpha}$ and $\hat{\beta}$ are selected values satisfying some optimality condition. An alternative, is to incorporate the hyperparameters in the posterior sampling scheme as delineated in Section 2.2.1. Finally, considering (only in this section) that β is known, one could also directly infer the parameters from Eq. (6) by using the so-called "true prior" $p(\boldsymbol{w}|\mathcal{H})$, which is given by integrating out the hyperparameter α based on its hyperprior, i.e.,

$$p(\boldsymbol{w}|\mathcal{H}) = \int p(\boldsymbol{w}|\alpha, \mathcal{H}) p(\alpha|\mathcal{H}) d\alpha \tag{56}$$

Note that the above calculation is made before any data has arrived.

A comparison of the optimization and integration alternatives in the context of Laplace approximation can be found in MacKay (1994). In summary, MacKay (1994) proposes to optimize the hyperparameters and to follow Eq. (55), because although integration leads to the correct maximum of the posterior $p(\boldsymbol{w}|\mathcal{D}, \mathcal{H})$, it does not capture its probability mass. However, this comparison is based on the Laplace approximation. More literature review is needed for a thorough comparison of these alternatives. In these notes the optimization and the sampling approaches are discussed; see also Section 2.3.4.

2.3.2 Evidence framework (empirical Bayes)

Suppose that in Eq. (18) $p(\alpha, \beta|\mathcal{H})$ is a hyperprior that assigns equal probability to all values of α, β; i.e., a non-informative prior. Then the hyperparameter posterior $p(\alpha, \beta|\mathcal{D}, \mathcal{H})$ is given up to a constant by $p(\mathcal{D}|\alpha, \beta, \mathcal{H})$, which is the evidence; note that $p(\mathcal{D}|\alpha, \beta, \mathcal{H})$ is also the denominator in Eq. (16). Therefore, a way of selecting α, β is by maximizing the evidence, i.e.,

$$\hat{\alpha}, \hat{\beta} = \underset{\alpha, \beta}{\operatorname{argmax}}\; p(\mathcal{D}|\alpha, \beta, \mathcal{H}) \tag{57}$$

where

$$p(\mathcal{D}|\alpha, \beta, \mathcal{H}) = \int p(\mathcal{D}|\boldsymbol{w}, \beta, \mathcal{H}) p(\boldsymbol{w}|\alpha, \mathcal{H}) d\boldsymbol{w} \tag{58}$$

This evidence maximization technique is widely used in Bayesian statistics and is also known as empirical Bayes or type II maximum likelihood estimation. See also MacKay (1995b) p. 35 where the evidence framework is compared with the minimum description length concept used in Hinton and Van Camp (1993) in conjunction with variational inference. In short, MacKay (1995b) claims that with care one can replicate Bayesian results in terms of minimum description lengths and sees no advantage of using the latter.

In this section some basic notions are introduced mainly based on the works of MacKay. The evidence may be hard to compute and thus hard to optimize the hyperparameters using it. In this regard, see Neal (1993) (Section 6) and Robert and Wraith

(2009) for surveys on computational methods, as well as Sohl-Dickstein and Culpepper (2012). Nevertheless, analytical expressions exist if the Laplace approximation is used for the posterior. Moreover, the evidence framework coupled with Laplace approximation gives us deep insights on effective dimensionality and generalization as noted in Maddox et al. (2020) and in Section 2.3.2.4. Finally, see Krishnan et al. (2019) for a more modern approach to Empirical Bayes.

2.3.2.1 Evidence in conjunction with Laplace approximation: According to Laplace approximation (Section 2.2.4) the parameter posterior can be obtained by Eq. (41). By substituting Eq. (41) into Eq. (58) an analytical expression (Eq. (4.6) in MacKay, 1992b) is obtained, which can be optimized numerically for obtaining $\hat{\alpha}, \hat{\beta}$ (see MacKay, 1992b and MacKay, 1992a). Specifically, optimization of the hyperparameters involves alternating the optimization of $\boldsymbol{w}$ (around which the Gaussian is fitted) for fixed hyperparameters with re-estimation of the hyperparameters by re-evaluating the Hessian matrix for the new value of $\boldsymbol{w}$ (see also Section 2.3.4). Error bars for the hyperparameters can be obtained by approximating the evidence maximum with a Gaussian distribution. See also MacKay (1995b), Sections 4.1 and 3.2, for modifications of the evidence framework for accounting for multimodal posteriors such as the ones arising in NNs and for multiple regularization constants, respectively.

2.3.2.2 Occam's razor: This is a principle of *parsimony of explanations*; between two similarly performing explanations, we should prefer the simpler one. In this regard, the evidence framework automatically incorporates this principle. This is illustrated in Fig. 8. An overcomplex model can model many different datasets, but it is unlikely that the specific dataset $\mathcal{D}$ was produced by that complex model (it has low evidence). Also a model that is too simple can model only a few out of all the possible datasets. Thus, the specific dataset $\mathcal{D}$ may have low evidence under such a model as well. As a result, the evidence framework incorporates automatically a trade-off between fitting the data and utilizing overcomplex models.

For an intuitive explanation, consider next Fig. 9, where $\boldsymbol{w}$ is one-dimensional ($k = 1$) and $\mathcal{H}_i$ denotes a specific value for the hyperparameter α, while β is considered known. For this case the evidence of Eq. (58) can be approximated as

$$p(\mathcal{D}|\alpha, \beta, \mathcal{H}) \approx p(\mathcal{D}|\hat{w}, \beta, \mathcal{H})p(\hat{w}|\alpha, \mathcal{H})\Delta w \tag{59}$$

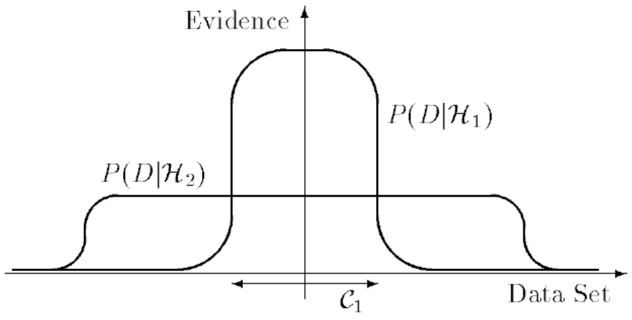


Fig. 8. Figure 2 from MacKay (1992b): Bayes favors more probable models. If the dataset falls in the region C_1 then the less powerful model $\mathcal{H}_1$ will be selected; $\mathcal{H}_1$ can capture a smaller number of datasets, but because the distribution must be normalized the evidence for those datasets will be larger.

where Δw is the width of the mode of the posterior. If the prior assigns equal probability to all values of w and has width $\Delta^0 w$, Eq. (59) becomes

$$p(\mathcal{D}|\alpha, \beta, \mathcal{H}) \approx p(\mathcal{D}|\hat{w}, \beta, \mathcal{H}) \frac{\Delta w}{\Delta^0 w} \tag{60}$$

where $\frac{\Delta w}{\Delta^0 w}$ is called Occam's factor. In summary, the evidence penalizes overcomplex models (with high prior variance $\Delta^0 w$), as well as models that need to shrink extremely in order to capture the data (too low posterior variance Δw; finely tuned to fit data). For a k-dimensional parameter vector $\boldsymbol{w}$, Eq. (59) becomes

$$p(\mathcal{D}|\alpha, \beta, \mathcal{H}) \approx p(\mathcal{D}|\hat{\boldsymbol{w}}, \beta, \mathcal{H}) p(\hat{\boldsymbol{w}}|\alpha, \mathcal{H}) (2\pi) det^{-1/2} \boldsymbol{A} \tag{61}$$

where $\boldsymbol{A}$ denotes the Hessian of the loss given as $\boldsymbol{A} = -\nabla\nabla \log p(\hat{\boldsymbol{w}}|\mathcal{D}, \hat{\alpha}, \hat{\beta}, \mathcal{H})$ and $p(\hat{\boldsymbol{w}}|\alpha, \mathcal{H})(2\pi)det^{-1/2}\boldsymbol{A}$ is the Occam's factor. In the multi-dimensional case, thus, the evidence framework penalizes overcomplex models, as well as posteriors having Hessian with large determinant value.

As mentioned in Maddox et al. (2020), more certainty about the parameters (say because of arrival of more data) leads to an increase in the curvature of the loss at the optimum. This increase in curvature of the loss leads to an increase in the eigenvalues of the Hessian of the loss. Specifically, there is a gap in the eigenspectrum of the Hessian: a small number of eigenvalues become large while the rest take on values near 0. Further, the directions with large eigenvalues are the ones that define effective dimensionality (see Section 2.3.2.3). Equivalently, as the eigenvalues of the Hessian

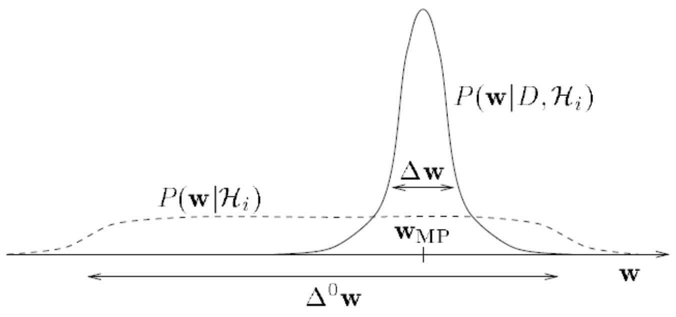

$P(\mathbf{w}|D, \mathcal{H}_i)$
$\Delta\mathbf{w}$
$P(\mathbf{w}|\mathcal{H}_i)$
$\mathbf{w}_{\mathrm{MP}}$
$\mathbf{w}$
$\Delta^0\mathbf{w}$

Fig. 9. Figure 3 from MacKay (1992b): The quantities that determine Occam's factor. Bayes penalizes too complex models with large $\Delta^0 w$ and shrunk posteriors with small Δw.

increase, the eigenvalues of the covariance matrix of the posterior distribution shrink, indicating contraction around the MAP estimate. Once again, Occam's razor penalizes overcomplex models (small prior variance), as well as posterior contraction. See Maddox et al. (2020) for efficient computation of the Hessian eigenvalues and Ghorbani et al. (2019) for a study on the Hessian eigenvalues in modern NNs.

Of course, the above illustrations pertain mainly to the Laplace approximation, but serve well for intuitively understanding the role of evidence. Most importantly, Maddox et al. (2020) relates effective dimensionality and how shrunk the posterior is with the generalization properties of NNs (see Section 2.3.2.4).

Overall, penalizing overcomplex models *arises naturally* in the Bayesian framework. Finally, it is worth pointing out that the variance of the prior does not necessarily affect the complexity of the output samples. Wilson and Izmailov (2020) shows that for RELU NNs modifying prior variances of the layers affects only the output scale, as opposed to sigmoid NNs considered in MacKay (1995b). This is an interesting topic that requires more literature review.

2.3.2.3 Effective dimensionality and generalization: As mentioned above, the hyperparameters can be optimized by numerically maximizing the evidence. As shown in MacKay (1992b) however, a deeper understanding of the problem is possible by considering the properties of the maximum evidence (see also Maddox et al., 2020). In general, given a dataset $\mathcal{D} = \{y_i\}_{i=1}^N$ of N Gaussian measurements with known mean

μ, the misfit written as

$$\chi^2_{\mathcal{D}} = \sum_{i=1}^{N} \frac{(y_i - \mu)^2}{\sigma^2_\epsilon} \tag{62}$$

follows a Chi-squared distribution with N degrees of freedom, because we have a sum of N independent random variables. Note that in these notes $\mathcal{D}$ is used for both measurements paired with their locations, i.e., $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$, and measurements alone, i.e., $\mathcal{D} = \{y_i\}_{i=1}^N$. In this regard, according to the discrepancy principle of statistics, we should estimate the noise level σ^2_ϵ by setting $\chi^2_{\mathcal{D}} = N$, i.e.,

$$\hat{\sigma}^2_\epsilon = \frac{\sum_i (y_i - \mu)^2}{N} \tag{63}$$

However, if the mean is also estimated by the data via $\hat{\mu} = \frac{\sum_i y_i}{N}$, then we have $N - 1$ independent random variables, i.e., $N - 1$ degrees of freedom in the Chi-squared distribution and the noise level should be estimated by $\chi^2_{\mathcal{D}} = N - 1$, i.e.,

$$\hat{\sigma}^2_\epsilon = \frac{\sum_i (y_i - \hat{\mu})^2}{N - 1} \tag{64}$$

In general, if k free parameters are determined by the data, then we have $N - k$ degrees of freedom and σ^2_ϵ is estimated by $\chi^2_{\mathcal{D}} = N - k$ (see MacKay, 1992b for discussion).

Following the same (frequentist) rationale, the noise level $\beta = 1/\sigma^2_\epsilon$ in regression should be estimated by $\mathcal{L}_{\mathcal{D}} = N - k$, because $\boldsymbol{w}$, which is k-dimensional, is estimated by the data. However, following the Bayesian approach of estimating the hyperparameters by maximizing the evidence, it can be shown that the resulting β estimate satisfies, instead, the condition $\mathcal{L}_{\mathcal{D}} = N - \gamma$, where $\gamma = k - \alpha Trace \boldsymbol{A}^{-1}$ represents the number of dimensions of $\boldsymbol{w}$ that have been well-determined by the data. Further, the "misfit" from the prior, denoted as $\mathcal{L}_{\boldsymbol{w}}$, is equal to $k - (k - \gamma) = \gamma$. In summary, only $\gamma < k$ parameters of $\boldsymbol{w}$ have been determined by the data, while $k - \gamma$ are determined by the prior (roughly set to zero and not contributing to reducing the training error).

Next, as stated in MacKay (1992a), sampling theory predicts that the addition of redundant parameters to a model should reduce $\mathcal{L}_{\mathcal{D}}$ by one unit per well-measured parameter. Thus, a stopping criterion could detect the point at which as parameters are deleted, $\mathcal{L}_{\mathcal{D}}$ starts to increase faster than with gradient 1 with decreasing γ (see Fig. 10). This could help us *select the number of hidden units.* Moody (1991) in a similar way penalizes large values of γ. In this regard, see Maddox et al. (2020) and Section 2.3.2.4 for connections between effective dimensionality, model selection and generalization behavior in modern deep learning.

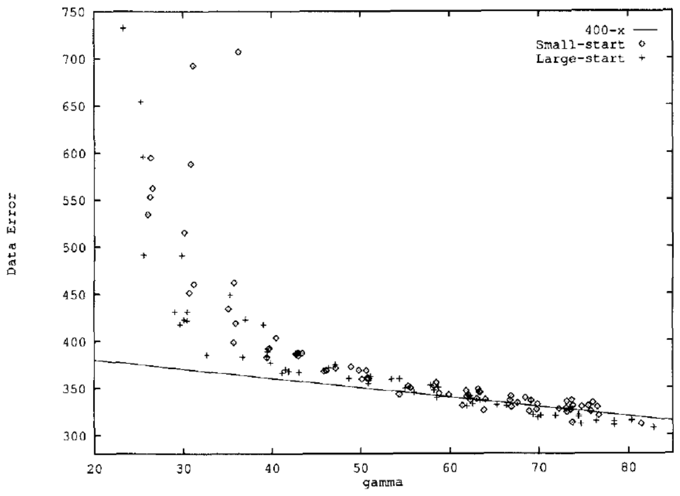


Fig. 10. Figure 6 from MacKay (1992a): Data misfit versus γ. Toward the right side of the figure, the data misfit is reduced by 1 for every well-determined parameter. When the model has too few parameters, however (toward the left of the figure), the misfit gets worse at a greater rate.

2.3.2.4 Rethinking parameter counting: Maddox et al. (2020) shows that effective dimensionality (having been obtained from *training data only*) can be a better proxy for generalization than naive parameter counting. Specifically, in over-parametrized cases ($k \gg N$) many dimensions of the posterior are left unchanged from the prior. As a result, the effective dimensionality is substantially less than the number of parameters k; we are unable to determine many more parameters than we have data observations. In this regard, Maddox et al. (2020) shows that when comparing different models, the ones with smaller effective dimensionality (less well-determined parameters) lead to better generalization and thus, should be selected. The latter is also supported by the fact that the Occam's factor in Eq. (61) does not only penalize models with too many parameters (these models have small $p(\hat{\boldsymbol{w}}|\alpha, \mathcal{H})$), but also models with shrunk posteriors (these models have small $det^{-1/2}\boldsymbol{A}$).

Further, a perturbation of of the MAP estimate $\hat{\boldsymbol{w}}$ in the undetermined directions (called degenerate directions in Maddox et al., 2020) gives almost identical functions (this is called function space homogeneity). As a result, effective dimensionality can be interpreted as model compression, since the undetermined directions do not contain additional functional information. Therefore, smaller effective dimensionality is interpreted as better model compression.

They show also that effective dimensionality can capture the phenomenon of double descent of the generalization performance (Nakkiran et al., 2019). In short, as the model grows, there exist a greater variety of subspaces which provide more effective compression of the data, and thus we achieve a lower effective dimensionality and better generalization properties. This is also related to the "blessing" of dimensionality described in Huang et al. (2019): flat regions of the loss occupy a greatly increasing volume and thus, are more easily discoverable by optimization. These solutions have lower effective dimensionality and thus, provide better lossless compression of the data (Maddox et al., 2020).

2.3.2.5 Should we optimize parameters and hyperparameters together in Laplace approximation? A point estimate of $\boldsymbol{w}$ could be obtained by maximizing $p(\mathcal{D}|\boldsymbol{w}, \alpha, \beta, \mathcal{H}) = p(\mathcal{D}|\boldsymbol{w}, \beta, \mathcal{H})p(\boldsymbol{w}|\alpha, \mathcal{H})$ with respect to $\boldsymbol{w}$, α and β simultaneously. Note that $p(\mathcal{D}|\boldsymbol{w}, \alpha, \beta, \mathcal{H})$ could be regarded as the likelihood of a model that considers $\boldsymbol{w}, \alpha$, and β as parameters and does not distinguish between hyperparameters and parameters. Next, this point estimate could be used either to make predictions directly using Eq. (11) or to fit a Gaussian distribution around it (see Section 2.2.4). However, as discussed in MacKay (1995a), this would lead to a biased estimate for the hyperparameters.

As an example, consider the case of fitting a Gaussian (with mean μ and variance σ_ϵ^2) to a dataset $\mathcal{D} = \{y_i\}_{i=1}^N$. The maximum likelihood estimator is given as $(\hat{\mu}, \hat{\sigma}_\epsilon) = (\frac{\sum_i y_i}{N}, \frac{\sum_i (y_i - \hat{\mu})^2}{N})$, which, in frequentist terms, is biased. As a result, $(\hat{\mu}, \hat{\sigma}_\epsilon) = (\frac{\sum_i y_i}{N}, \frac{\sum_i (y_i - \hat{\mu})^2}{N-1})$ should be used instead. In Bayesian terms, since we are interested in maximizing the evidence $p(\mathcal{D}|\sigma_\epsilon)$, we should average over all possible values of μ and then optimize for σ_ϵ. Equivalently, we should optimize $p(\mathcal{D}|\sigma_\epsilon)$ and not $p(\mathcal{D}|\sigma_\epsilon, \mu = \hat{\mu})$. An approximation of $p(\mathcal{D}|\sigma_\epsilon)$ (or equivalently of Eq. (58)) is what the evidence framework optimizes in Eq. (57).

Overall, in practice either an iterative approach is used that alternates between parameter inference and hyperparameter optimization, or parameter inference is performed for different values of hyperparameters and the optimal parameters are selected based on some condition (see Section 2.3.4).

2.3.2.6 Model comparison: In the same way we compare between different hyperparameter settings by comparing the resulting evidence, we can compare between different models $\mathcal{H}$. Suppose, for example, we would like to compare between two models with different types of priors. This is possible by comparing $p(\mathcal{D}|\mathcal{H}_i)$ for different values of i corresponding to different models/architectures. See for example Section 6 in MacKay (1992b), where the error bars of the optimized hyperparameters are used for obtaining $p(\mathcal{D}|\mathcal{H}_i)$. Also see Rasmussen and Ghahramani (2001), where it is claimed that automatic Occam's razor, i.e., penalization of overcomplex models, is present in both selecting hyperparameters and models. In both cases, evidence penalizes *overcomplex functions* implied by the functional prior.

2.3.2.7 Evidence in conjunction with variational inference: As with Laplace approximation, optimizing the hyperparameters for the case of approximate inference by variational inference involves alternating maximization of the ELBO with respect to the posterior distribution q for fixed hyperparameters and with respect to the hyperparameters for fixed q.

2.3.3 Cross-validation

In standard training of NNs cross-validation (k-fold typically) is predominantly used for tuning the regularization parameters. A review of cross-validation techniques for model selection can be found in Arlot and Celisse (2010). In this section its relation with evidence is briefly reviewed.

2.3.3.1 Relation with evidence: The evidence of a particular data set $\mathcal{D} = \{y_i\}_{i=1}^N$ for a model $\mathcal{H}$ can be written as

$$p(y_1, \ldots, y_N|\mathcal{H}) = p(y_1|\mathcal{H})p(y_2|y_1, \mathcal{H}) \cdots p(y_N|y_1, \ldots, y_{N-1}, \mathcal{H}) \tag{65}$$

where the specific permutation of data indices does not matter. Equivalently, the log evidence can be written as a sum of N log conditionals. Each of the N terms is different depending on the particular permutation, but the sum is the same. The leave-one-out cross-validation error is $-\log p(y_N|y_1, \ldots, y_{N-1}, \mathcal{H})$ and it is a noisy error because it depends on the specific permutation. This is why in cross-validation the average is taken and a large data set may be required to reduce the signal to noise ratio. Further, Fong and Holmes (2020) shows that the evidence is N times the average leave-p-out

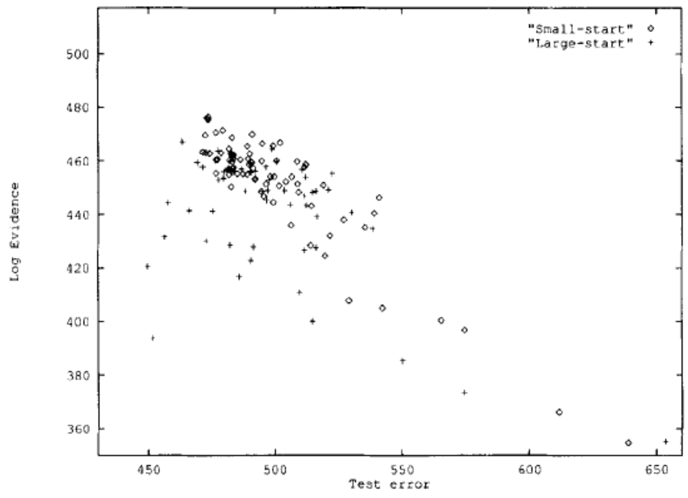


Fig. 11. Figure 7 from MacKay (1992a): The desired correlation between evidence and test error has negative slope. However, a significant number of points on the lower left side of the figure violate this desired trend.

cross-validation score, where the average is taken with respect to p.

MacKay (1992b) suggests that leave-one-out cross-validation is a good predictor of $-\log p(y_{N+1}|y_1, \ldots, y_N, \mathcal{H})$, i.e., of the generalization error, for large values of N. However, one may not want to sacrifice useful data as a test set. In addition, cross-validation may be computationally demanding and cannot easily handle many hyperparameters. Finally, a large test set may be required to reduce the signal to noise ratio.

On the other hand, the evidence may not be well-correlated with the generalization error (see Fig. 11); it is after all the area under the curve and may not be representative of the tail of the curve. If this happens we can gain useful insight into defects of the model, e.g., the regularizer may not be appropriate. In this regard, MacKay (1992a) shows that by introducing separate hyperparameters for each layer the correlation with the generalization error is improved. In addition, in p. 457 MacKay (1992a) suggests that γ may serve as a useful tool for generalization error (see also Maddox et al., 2020 and Section 2.3.2.4 herein). For this reason, MacKay (1992a) proposes to do both, cross-validation and evidence computation for assessing the generalization properties of the BNN.

2.3.4 Alternating between approximate inference and hyperparameter optimization

In conclusion, either a two-step approach is followed, i.e., approximate inference and then hyperparameter optimization, or an iterative approach, i.e., alternation between approximate inference and hyperparameter optimization. As pointed out by Neal and Hinton (1998), there is a relation between the variational framework (variational inference combined with hyperparameter optimization) and the expectation maximization (EM) algorithm by regarding the parameter vector $\boldsymbol{w}$ as a latent variable. Specifically, the E-step corresponds to ELBO maximization with respect to q (Section 2.2.5) and the M-step corresponds to optimization of the hyperparameters (see also Barber and Bishop, 1998 for discussion, Gal, 2016 p. 25, as well as Jospin et al., 2020 algorithm 3).

An alternative technique in the spirit of alternation is using Gibbs sampling steps to update the hyperparameters as done in Neal (1995) and Nalisnick (2018). Specifically, Neal (1995) does not optimize the hyperparameters, but attempts to approximate the predictive distribution via Eq. (19) by obtaining samples from $p(\boldsymbol{w}, \alpha, \beta | \mathcal{D}, \mathcal{H})$ instead of $p(\boldsymbol{w}|\mathcal{D}, \hat{\alpha}, \hat{\beta}, \mathcal{H})$ for optimal values $\hat{\alpha}, \hat{\beta}$. This is done by a "global" Gibbs sampling scheme (Section 2.2.3.2): The parameters $\boldsymbol{w}$ are updated from the conditional $p(\boldsymbol{w}|\mathcal{D}, \alpha, \beta, \mathcal{H})$ via HMC (Section 2.2.3.5) for given hyperparameters α, β and then α, β are drawn from $p(\alpha, \beta|\mathcal{D}, \boldsymbol{w}, \mathcal{H})$ given the last update of the parameters $\boldsymbol{w}$. See, for example, Eq. (3.17) in Neal (1995) for an indicative form of the conditional $p(\alpha, \beta|\mathcal{D}, \boldsymbol{w}, \mathcal{H})$.

2.4 Uncertainty quality

As discussed in Sections 1.2.1-1.2.2, BNNs calibrate the predictions of NNs by incorporating epistemic uncertainty. However, Foong et al. (2019a) notes that BNNs have yet to become mainstream in deep learning, in part due to concerns raised regarding the quality of this uncertainty. Precisely, the BNN output is obtained by the approximate inference techniques described in Section 2.2, which have an effect on the uncertainty quality.

We need metrics for measuring the quality of the obtained uncertainty, as well as methods for improving it. One aspect of uncertainty quality is *calibration*. Calibration is measured typically with scoring rules, briefly discussed in Lakshminarayanan et al. (2017). Another aspect is *sharpness*; in a regression context, this means that the confi-

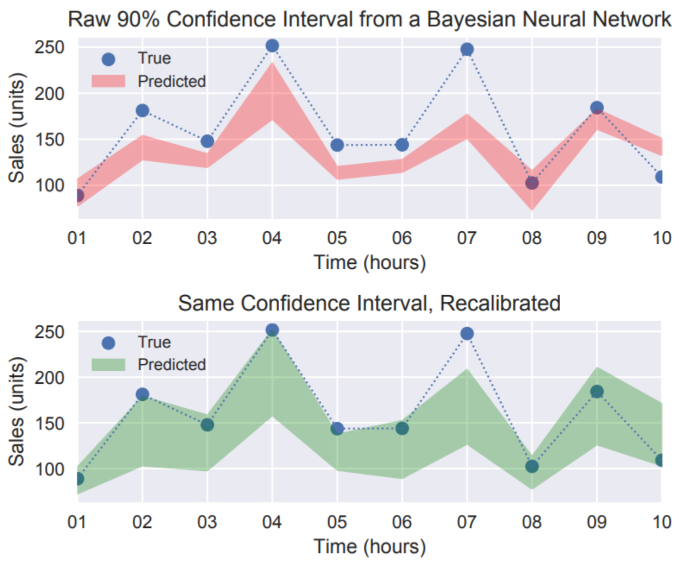


Fig. 12. Figure 1 from Kuleshov et al. (2018): Top: Time series forecasting using a Bayesian neural network. Because the model is probabilistic, a 90% credible interval around the forecast can be obtained (red). However, the interval fails to capture the true data distribution: most points fall outside of it. Bottom: The recalibration method proposed in Kuleshov et al. (2018) enables the original model to output a 90% credible interval (green) that correctly contains 9/10 points.

dence intervals should all be as tight as possible around a single value (Kuleshov et al., 2018). Further, improving the obtained uncertainty is related to *recalibration*, discussed in Sections 1.2.1 and 2.4.3.

2.4.1 In-domain uncertainty

In-domain uncertainty of the model is measured on data taken from the training data distribution, i.e. data from the same domain. Kuleshov et al. (2018) notes that, in practice, Bayesian uncertainty estimates often fail to capture the true data distribution; e.g., a 90% posterior credible interval generally does not contain the true outcome 90% of the time (Fig. 12).

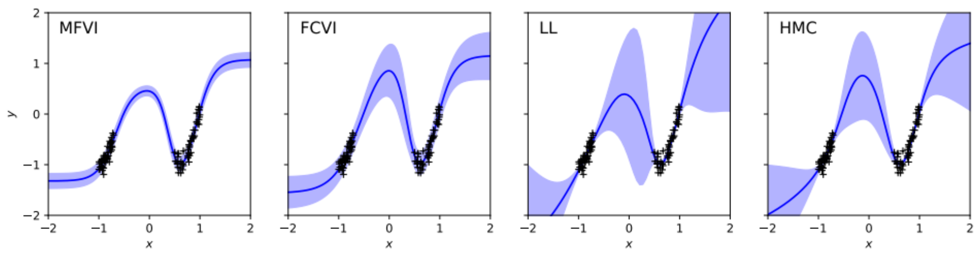


Fig. 13. Figure 1 from Foong et al. (2019b): Mean and two standard deviation bars of the predictive distribution obtained with MFVI, full covariance VI (see Section 2.2.5), Laplace approximation and HMC.

2.4.2 In-between uncertainty

Out-of-domain or out-of-distribution uncertainty of the model is measured on the data that does not follow the same distribution as the training dataset (out-of-distribution data). An example of out-of-distribution data is the region located in-between regions that contain training samples (Fig. 13). Foong et al. (2019b) shows that mean field variational inference (VI of Section 2.2.5 with fully factorized $q_{\boldsymbol{\theta}}(\boldsymbol{w})$; MFVI) fails to represent in-between uncertainty: its error bars are of similar magnitude in the data region and the in-between region, thus it is overconfident (Fig. 13). The same authors provide theoretical results in Foong et al. (2019a). In short, they show how the uncertainty quality depends on the expressiveness of $q_{\boldsymbol{\theta}}(\boldsymbol{w})$.

2.4.3 Recalibration of regression BNNs

In general, for some input value x_t the BNN outputs a cumulative distribution function (CDF) $F_{x_t}(y)$, or for notation simplicity, $F_t(y)$. This CDF is related to $p(y^*|x^*, \mathcal{D}, \mathcal{H})$ in Eq. (3), but the notation of Kuleshov et al. (2018) is followed in this section. The BNN is calibrated if

$$\frac{\sum_{t=1}^{T} \mathbf{1}\{y_t \leq F_t^{-1}(p)\}}{T} \to p \tag{66}$$

for all $p \in [0, 1]$, as $T \to \infty$. In other words, the empirical and the predicted CDFs should match as the dataset size goes to infinity. Note that the above definition does not distinguish between different locations x_t; the BNN is *on average* calibrated. When the values x_t, y_t are i.i.d. realizations of random variables X, Y coming from the data-generating distribution, then a sufficient condition for calibration is

$$P(Y \leq F_X^{-1}(p)) = p \tag{67}$$

Note that Eq. (67) is similar to Eq. (20). Again, this is a probability over the random variable X that denotes the input in general, and not a specific x location.

Kuleshov et al. (2018) proposes to train an auxiliary model $R : [0,1] \to [0,1]$, which takes as input the CDF F_t and produces a recalibrated one, i.e., $R \circ F_t$. This can be done by estimating from data the probability $P(Y \leq F_X^{-1}(p))$. As an example, consider that for $x_t = 3$ the output F_t predicts $F_t(20) = 0.95$. This means that the 95% quantile is $F_t^{-1}(0.95) = 20$. If, however, our data-based estimate for $P(Y \leq F_X^{-1}(p))$ predicts that $P(Y < 20) = 0.8$, then we set $F_t(20) = 0.8$. We can do that for a number of quantiles, e.g., for $5\%, 10\%$, etc. The effect of recalibration is shown in Fig. 12. Criticism and extensions of this approach can be found in Levi et al. (2019), Song et al. (2019), Cui et al. (2020), and Zelikman et al. (2020).

2.4.4 Diagnostic tools

2.4.4.1 Calibration: According to Kuleshov et al. (2018), a calibration plot for regression similar to the one for classification (Fig. 1) can be constructed. We divide the confidence axis into M intervals and for each one we compute the empirical frequency

$$\hat{p}_m = \frac{|\{y_t | F_t(y_t) \leq p_m, t = 1, \ldots, T\}|}{T} \tag{68}$$

and plot $\{(p_m, \hat{p}_m)\}_{m=1}^{M}$. Calibrated forecasts correspond to a straight line. For best results, the diagnostic dataset should be distinct from the calibration and training sets. We can also obtain a single numerical score describing the quality of forecast calibration given by

$$score = \sum_{m=1}^{M} (p_m - \hat{p}_m)^2 \tag{69}$$

2.4.4.2 Sharpness: Sharpness can be measured using the variance of the predictions. Low-variance predictions are tightly centered around one value. A sharpness score can be defined by

$$sharpness = \frac{1}{T} \sum_{t=1}^{T} var(F_t) \tag{70}$$

i.e., the mean output variance of the members of the diagnostic set.

References

Arlot, S. and Celisse, A. (2010). "A Survey of Cross-Validation Procedures for Model Selection". *Statistics surveys* 4, pp. 40–79.

Ashukha, A., Lyzhov, A., Molchanov, D., and Vetrov, D. (2020). "Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning". *arXiv:2002.06470 [cs, stat]*. arXiv: 2002.06470 [cs, stat].

Barber, D. and Bishop, C. M. (1998). "Ensemble Learning in Bayesian Neural Networks". *Nato ASI Series F Computer and Systems Sciences* 168, pp. 215–238.

Betancourt, M. (2017). "A Conceptual Introduction to Hamiltonian Monte Carlo". *arXiv preprint arXiv:1701.02434*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blei, D. M. (2014). "Build, Compute, Critique, Repeat: Data Analysis with Latent Variable Models".

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). "Weight Uncertainty in Neural Networks". *arXiv preprint arXiv:1505.05424*.

Box, G. E. P. and Tiao, G. C. (1973). *Bayesian Inference in Statistical Analysis*. Addison-Wesley Pub. Co.

Chen, T., Fox, E., and Guestrin, C. (2014). "Stochastic Gradient Hamiltonian Monte Carlo". *International Conference on Machine Learning*, pp. 1683–1691.

Cui, P., Hu, W., and Zhu, J. (2020). "Calibrated Reliable Regression Using Maximum Mean Discrepancy". *arXiv preprint arXiv:2006.10255*.

Denker, J. S. and LeCun, Y. (1991). "Transforming Neural-Net Output Levels to Probability Distributions". *Advances in Neural Information Processing Systems*, pp. 853–859.

Flam-Shepherd, D., Requeima, J., and Duvenaud, D. (2017). "Mapping Gaussian Process Priors to Bayesian Neural Networks". *NIPS Bayesian Deep Learning Workshop*.

Fong, E. and Holmes, C. (2020). "On the Marginal Likelihood and Cross-Validation". *Biometrika* 107, pp. 489–496.

Foong, A. Y., Burt, D. R., Li, Y., and Turner, R. E. (2019a). "On the Expressiveness of Approximate Inference in Bayesian Neural Networks". *arXiv*, arXiv–1909.

Foong, A. Y., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. (2019b). "'In-Between'Uncertainty in Bayesian Neural Networks". *arXiv preprint arXiv:1906.11537*.

Gal, Y. (2016). “Uncertainty in Deep Learning”. *University of Cambridge* 1.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. (2018). “Loss Surfaces, Mode Connectivity, and Fast Ensembling of Dnns”. *Advances in Neural Information Processing Systems*, pp. 8789–8798.

Ghorbani, B., Krishnan, S., and Xiao, Y. (2019). “An Investigation into Neural Net Optimization via Hessian Eigenvalue Density”. *arXiv preprint arXiv:1901.10159*.

Graves, A. (2011). “Practical Variational Inference for Neural Networks”. *Advances in Neural Information Processing Systems*, pp. 2348–2356.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). “On Calibration of Modern Neural Networks”. *arXiv preprint arXiv:1706.04599*.

Hafner, D., Tran, D., Lillicrap, T., Irpan, A., and Davidson, J. (2018). “Noise Contrastive Priors for Functional Uncertainty”. *arXiv preprint arXiv:1807.09289*.

Hinton, G. E. and Van Camp, D. (1993). “Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights”. *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pp. 5–13.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). “Stochastic Variational Inference”. *The Journal of Machine Learning Research* 14, pp. 1303–1347.

Huang, G. et al. (2017). “Snapshot Ensembles: Train 1, Get M for Free”. *arXiv:1704.00109 [cs]*. arXiv: `1704.00109 [cs]`.

Huang, W. R. et al. (2019). “Understanding Generalization through Visualizations”. *arXiv preprint arXiv:1906.03291*.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. (2019). “Averaging Weights Leads to Wider Optima and Better Generalization”. *arXiv:1803.05407 [cs, stat]*. arXiv: `1803.05407 [cs, stat]`.

Jospin, L. V., Buntine, W., Boussaid, F., Laga, H., and Bennamoun, M. (2020). “Hands-on Bayesian Neural Networks–a Tutorial for Deep Learning Users”. *arXiv preprint arXiv:2007.06823*.

Kendall, A. and Gal, Y. (2017). “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?” *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 5574–5584.

Kingma, D. P., Salimans, T., and Welling, M. (2015). “Variational Dropout and the Local Reparameterization Trick”. *Advances in Neural Information Processing Systems*, pp. 2575–2583.

Krishnan, R., Subedar, M., and Tickoo, O. (2019). “Specifying Weight Priors in Bayesian Deep Neural Networks with Empirical Bayes”. *arXiv:1906.05323 [cs, stat]*. arXiv: 1906.05323 [cs, stat].

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. M. (2017). “Automatic Differentiation Variational Inference”. *The Journal of Machine Learning Research* 18, pp. 430–474.

Kuleshov, V., Fenner, N., and Ermon, S. (2018). “Accurate Uncertainties for Deep Learning Using Calibrated Regression”. *arXiv preprint arXiv:1807.00263*.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). “Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles”. *Advances in Neural Information Processing Systems*, pp. 6402–6413.

Lee, J. et al. (2017). “Deep Neural Networks as Gaussian Processes”. *arXiv preprint arXiv:1711.00165*.

Levi, D., Gispan, L., Giladi, N., and Fetaya, E. (2019). “Evaluating and Calibrating Uncertainty Prediction in Regression Tasks”. *arXiv preprint arXiv:1905.11659*.

Louizos, C., Shi, X., Schutte, K., and Welling, M. (2019). “The Functional Neural Process”. *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., pp. 8746–8757.

Ma, C., Li, Y., and Hernández-Lobato, J. M. (2019). “Variational Implicit Processes”. *International Conference on Machine Learning*, pp. 4222–4233.

MacKay, D. J. (1992a). “A Practical Bayesian Framework for Backpropagation Networks”. *Neural computation* 4, pp. 448–472.

— (1992b). “Bayesian Interpolation”. *Neural computation* 4, pp. 415–447.

— (1994). “Hyperparameters: Optimise or Integrate Out”. *Maximum entropy and Bayesian methods*.

— (1995a). “Developments in Probabilistic Modelling with Neural Networks—Ensemble Learning”. *Neural Networks: Artificial Intelligence and Industrial Applications*. Springer, pp. 191–198.

— (1995b). “Probable Networks and Plausible Predictions—a Review of Practical Bayesian Methods for Supervised Neural Networks”. *Network: computation in neural systems* 6, pp. 469–505.

— (1998). “Introduction to Gaussian Processes”. *NATO ASI Series F Computer and Systems Sciences* 168, pp. 133–166.

Maddox, W. J., Benton, G., and Wilson, A. G. (2020). “Rethinking Parameter Counting in Deep Models: Effective Dimensionality Revisited”. *arXiv preprint arXiv:2003.02139*.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. (2019). “A Simple Baseline for Bayesian Uncertainty in Deep Learning”. *Advances in Neural Information Processing Systems*, pp. 13153–13164.

Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). “Stochastic Gradient Descent as Approximate Bayesian Inference”. *The Journal of Machine Learning Research* 18, pp. 4873–4907.

Matthews, A. G. d. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. (2018). “Gaussian Process Behaviour in Wide Deep Neural Networks”. *arXiv:1804.11271 [cs, stat]*. arXiv: 1804.11271 [cs, stat].

Mescheder, L., Nowozin, S., and Geiger, A. (2018). “Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks”. *arXiv:1701.04722 [cs]*. arXiv: 1701.04722 [cs].

Moody, J. E. (1991). “Note on Generalization, Regularization and Architecture Selection in Nonlinear Learning Systems”. *Neural Networks for Signal Processing Proceedings of the 1991 IEEE Workshop*. IEEE, pp. 1–10.

Nakkiran, P. et al. (2019). “Deep Double Descent: Where Bigger Models and More Data Hurt”. *arXiv preprint arXiv:1912.02292*.

Nalisnick, E. T. (2018). “On Priors for Bayesian Neural Networks”. PhD thesis. UC Irvine.

Neal, R. M. (1993). *Probabilistic Inference Using Markov Chain Monte Carlo Methods*. Department of Computer Science, University of Toronto Toronto, Ontario, Canada.

— (1995). “Bayesian Learning for Neural Networks”. PhD thesis. University of Toronto.

Neal, R. M. and Hinton, G. E. (1998). “A View of the EM Algorithm That Justifies Incremental, Sparse, and Other Variants”. *Learning in Graphical Models*. Springer, pp. 355–368.

Ovadia, Y. et al. (2019). “Can You Trust Your Model’s Uncertainty? Evaluating Predictive Uncertainty under Dataset Shift”. *Advances in Neural Information Processing Systems*, pp. 13991–14002.

Pang, G., Yang, L., and Karniadakis, G. E. (2019). “Neural-Net-Induced Gaussian Process Regression for Function Approximation and PDE Solution”. *Journal of Computational Physics* 384, pp. 270–288.

Rasmussen, C. E. and Ghahramani, Z. (2001). "Occam's Razor". *Advances in Neural Information Processing Systems*, pp. 294–300.

Ritter, H., Botev, A., and Barber, D. (2018). "A Scalable Laplace Approximation for Neural Networks". *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings.* Vol. 6. International Conference on Representation Learning.

Robert, C. P. and Wraith, D. (2009). "Computational Methods for Bayesian Model Choice". *Aip Conference Proceedings.* Vol. 1193. American Institute of Physics, pp. 251–262.

Simpson, D. P., Rue, H., Martins, T. G., Riebler, A., and Sørbye, S. H. (2015). "Penalising Model Component Complexity: A Principled, Practical Approach to Constructing Priors". *arXiv:1403.4630 [stat].* arXiv: 1403.4630 [stat].

Sohl-Dickstein, J. and Culpepper, B. J. (2012). "Hamiltonian Annealed Importance Sampling for Partition Function Estimation". *arXiv:1205.1925 [physics].* arXiv: 1205.1925 [physics].

Song, H., Diethe, T., Kull, M., and Flach, P. (2019). "Distribution Calibration for Regression". *arXiv preprint arXiv:1905.06023.*

Sun, S., Zhang, G., Shi, J., and Grosse, R. (2019). "Functional Variational Bayesian Neural Networks". *arXiv:1903.05779 [cs, stat].* arXiv: 1903.05779 [cs, stat].

Tishby, N., Levin, E., and Solla, S. A. (1989). "Consistent Inference of Probabilities in Layered Networks: Predictions and Generalization". *International Joint Conference on Neural Networks.* Vol. 2, pp. 403–409.

Welling, M. and Teh, Y. W. (2011). "Bayesian Learning via Stochastic Gradient Langevin Dynamics". *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 681–688.

Wenzel, F. et al. (2020). "How Good Is the Bayes Posterior in Deep Neural Networks Really?" *arXiv preprint arXiv:2002.02405.*

Williams, C. K. (1997). "Computing with Infinite Networks". *Advances in Neural Information Processing Systems*, pp. 295–301.

Williams, P. M. (1995). "Bayesian Regularization and Pruning Using a Laplace Prior". *Neural computation* 7, pp. 117–143.

Wilson, A. G. and Izmailov, P. (2020). "Bayesian Deep Learning and a Probabilistic Perspective of Generalization". arXiv: 2002.08791.

Yang, W. et al. (2019). “Output-Constrained Bayesian Neural Networks”. *arXiv preprint arXiv:1905.06287*.

Zelikman, E., Healy, C., Zhou, S., and Avati, A. (2020). “CRUDE: Calibrating Regression Uncertainty Distributions Empirically”. *arXiv:2005.12496 [cs, stat]*. arXiv: 2005.12496 [cs, stat].