# Hessian-Free Laplace in Bayesian Deep Learning

**James McInerney**
*Netflix Research*
JMCINERNEY@NETFLIX.COM

**Nathan Kallus**
*Cornell University & Netflix Research*
KALLUS@CORNELL.EDU

## Abstract

The Laplace approximation (LA) of the Bayesian posterior is a Gaussian distribution centered at the maximum *a posteriori* estimate. Its appeal in Bayesian deep learning stems from the ability to quantify uncertainty post-hoc (i.e., after standard network parameter optimization), the ease of sampling from the approximate posterior, and the analytic form of model evidence. Uncertainty in turn can direct experimentation. However, an important computational bottleneck of LA is the necessary step of calculating and inverting the Hessian matrix of the log posterior. The Hessian may be approximated in a variety of ways, with quality varying with a number of factors including the network, dataset, and inference task. In this paper, we propose an alternative algorithm that sidesteps Hessian calculation and inversion. The *Hessian-free Laplace* (HFL) approximation uses curvature of both the log posterior and network prediction to estimate its variance. Two point estimates are required: the standard maximum *a posteriori* parameters and the optimal parameter under a loss regularized by the network prediction. We show that under standard assumptions of LA in Bayesian deep learning, HFL targets the same variance as LA, and this is empirically explored in small-scale simulated experiments comparing against the exact Hessian.

**Keywords:** Bayesian Neural Networks, Laplace Approximation, Epistemic Uncertainty

## 1. Introduction

Bayesian neural networks, where a prior is placed on model weights, offer an opportunity to understand uncertainty and direct experimentation. However, exact computation of the model-weight posterior and predictive distributions is prohibitive given size and complexity. It is instead appealing to approximate it using the maximum *a posteriori* (MAP) solution and the curvature thereat. However, even this curvature, characterized by a Hessian, can be challenging to compute.

Consider the maximum *a posteriori* (MAP) solution of a Bayesian neural network $f_\theta$ with parameters $\theta$ given data $\{(x_i, y_i)\}_{i=1}^n$,

$$\hat{\theta} = \arg_\theta \max \mathcal{L}_\theta \tag{1}$$

$$\text{where } \mathcal{L}_\theta = \sum_{i=1}^{n} \log p(y_i \mid f_\theta(x_i)) + \log p(\theta), \tag{2}$$

for some observation likelihood $p(y \mid \phi)$ and prior $p(\theta)$. This is equivalent to optimizing a loss (e.g., squared loss for Gaussian likelihood or cross-entropy for categorical likelihood)

regularized by $\log p(\theta)$ and means that computing the MAP of a Bayesian neural network is akin to fitting a vanilla neural network.

As detailed in the next section, the Laplace approximation of of the posterior distribution leverages the (negative) Hessian of the log joint distribution at the MAP (Mackay, 1992),

$$P = -\nabla\nabla_\theta \mathcal{L}_{\hat{\theta}}. \tag{3}$$

Calculating and inverting $P$ are the central steps in performing the Laplace approximation. For high dimensional $\theta$ – common in deep neural networks – these steps are a prohibitive computational bottleneck even for an approximate posterior. One remedy is further approximation of the Hessian as well, but these may ignore important directions of curvature.

In this paper we propose Hessian-free Laplace (HFL), which directly approximates the uncertainty of the predicted output mean in the Laplace approximation without explicitly computing and inverting the Hessian (while still assuming it exists as in Laplace). Instead, one additional point estimate is required, the *prediction-regularized* MAP, derived from the network objective with a small amount of regularization added that is conditional on the query point $x$ of the prediction. Under the assumptions that Laplace approximation is used, we find that the rescaled difference in network outputs given by the MAP and the prediction-regularized MAP recovers the Laplace variance.

The rest of the paper is organized as follows. In Section 2, we develop the Hessian-free Laplace method. Empirical evaluation is presented in Section 3 before discussing conclusions and future work in Section 4.

## 2. Bayesian Deep Learning with Hessian-Free Laplace

Our starting point is the learning of parameters $\theta$ of a deep neural network $f_\theta$ as per Eq. 1 and 2. While the representational power and accuracy of the network may be high, in many applications it is crucial to also quantify the uncertainty of predictions, such as in autonomous vehicles, healthcare, risk-averse recommendations (Kendall and Gal, 2017; Leibig et al., 2017).

Bayesian probability theory provides a framework for analyzing uncertainty in deep learning, referred to as Bayesian deep learning (BDL). As for Bayesian inference in general, the key components are the prior distribution $p(\theta)$ and likelihood function $p(y \mid \theta, x)$ in a family of models indexed by $\theta$, and an i.i.d. dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ assumed to be collected from a model in this family. The posterior distribution is then derived,

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{\int p(\mathcal{D} \mid \theta)p(\theta)\mathrm{d}\theta}, \tag{4}$$

and may be flexibly used as the density (or mass) for the expectation of any downstream prediction depending on $\theta$, e.g. expectation or variance of predicted mean, as part of a larger simulation involving fitted deep models, or interpreting the variance of weights for different layers of the network. The main evaluations of interest in BDL are the mean $\mathbb{E}_{p(\theta \mid \mathcal{D})}[f_\theta(x)]$ and (co)variance $\mathrm{Var}_{p(\theta \mid \mathcal{D})}[f_\theta(x)]$ of the prediction for a query vector (or set of query vectors) $x$. These will be the focus of our study in this paper.

**Double Intractability** The steps in BDL discussed so far are doubly intractable in the following ways.

- **Model Evidence** For all but the most trivial networks, the integral in the denominator of Eq. 4, (the model evidence) entails a sum over high-dimensional $\theta$. Many techniques have been developed to approximate the posterior – the most common being MAP estimation, variational inference, and Markov chain Monte Carlo – and these vary in their scalability and applicability to deep neural nets (Blundell et al., 2015; Ritter et al., 2018). A key desideratum for approximate inference that is often overlooked is the extent to which it is compatible with existing deep learning frameworks and implementations, which may be supported by multiple years of model selection, tuning, and vast training budgets. The MAP point estimate is the most practical in this sense, and is the basis of the Laplace approximation.

- **Posterior Predictive** The posterior predictive $p(y \mid \mathcal{D}, x)$ is the expected predictive density, involving complex function $f_\theta$, under averages of the approximate posterior, meaning that the integral cannot be further simplified. Monte Carlo estimation using samples from the posterior is used to address this intractability.

**Laplace Approximation** Let $q$ be the Laplace approximation of the posterior, defined as the second-order Taylor expansion around $\hat{\theta}$,

$$\log q(\theta) = \log p(\hat{\theta} \mid \mathcal{D}) - \frac{1}{2}(\theta - \hat{\theta})^\top P(\theta - \hat{\theta}), \tag{5}$$

where $P \mathrel{\widehat{=}} -\nabla\nabla_\theta \log p(\theta \mid \mathcal{D})$. The first derivative does not appear in Eq. 5 because $\hat{\theta}$ is defined as a local optimum of the posterior, where it exists.

By inspection of Eq. 5, we see that,

$$q(\theta) \propto \exp\left\{-\frac{1}{2}(\theta - \hat{\theta})^\top P(\theta - \hat{\theta})\right\} \tag{6}$$

$$\implies q(\theta) = \mathcal{N}(\hat{\theta}, P^{-1}). \tag{7}$$

If precision matrix $P$ can be calculated and inverted, then the first intractability of approximating the posterior is addressed. In addition, Eq. 7 also provides a closed-form solution to the model evidence, supporting model and hyperparameter selection (MacKay, 1992). We next discuss precision calculation in more detail.

**Precision Matrix** Eq. 7 depends on calculating the covariance matrix, i.e. inverting the precision matrix $P$ which can be decomposed as,

$$\begin{aligned} P &= -\nabla\nabla_\theta \log p(\theta \mid \mathcal{D}) \\ &= \underbrace{-\nabla\nabla_\theta \log p(\theta)}_{\text{Gaussian prior} \implies \text{scalar matrix}} \underbrace{-\nabla\nabla_\theta \log p(\mathcal{D} \mid \theta)}_{\text{Hessian } H}. \end{aligned} \tag{8}$$

The prior term in Eq. 8 usually takes a simple form, e.g. constant scalar for Gaussian prior, so our focus from this point will be on $H$, the Hessian of the *likelihood*.

The Hessian of the likelihood term may be further decomposed as,

$$\begin{aligned} H &= \nabla\nabla_\theta \log p(y \mid f_\theta(x)) \\ &= \underbrace{\nabla_f \log p(y \mid f_\theta(x))}_{\text{residual}} \nabla\nabla_\theta f_\theta(x) + \underbrace{\nabla\nabla_f \log p(y \mid f_\theta(x))}_{\text{observation precision}} \nabla_\theta f_\theta(x) \nabla_\theta f_\theta(x)^\top. \end{aligned} \tag{9}$$

In practice, the precision calculation of the Laplace approximation drops the first term – the network Hessian – in Eq. 9 due to computational constraints. This is known as generalized Gauss-Newton (GGN) and is proportional to the outer product of the network Jacobian (second term in Eq. 9). Beyond expediency, there are two arguments motivating GGN:

1. When fitting highly flexible models, as in BDL, the residual error is expected to be $\approx 0$.

2. The residual is a zero-mean random variable uncorrelated with the network Hessian, so the first term is approximately zero as the number of data points grows.

Immer et al. (2021) observed that when GGN is used in approximate inference, it assumes the network takes the form of a generalized linear model. In Laplace, this is equivalent to a local linearization of the network prediction around $\hat{\theta}$,

$$\tilde{f}_\theta(x) \mathrel{\widehat{=}} f_{\hat{\theta}}(x) + \nabla_\theta f_\theta(x)\Big|_{\hat{\theta}}^\top (\theta - \hat{\theta}), \tag{10}$$

which experimentally results in more accurate predictions than the original network $f_{\hat{\theta}}$ when used in the Monte Carlo average for evaluation (Foong et al., 2019). However, the Hessian, even after simplifying with GGN, is still unwieldy or prohibitive to compute, store, and invert for large networks with high dimensional $\theta$.

**Hessian-free Laplace (HFL)** Against this background, we target the same mathematical object as the Laplace approximation, requiring the Hessian to exist but avoiding the need to calculate or invert it. To start, notice that an immediate consequence of Eq. 10 is,

$$\tilde{f}_\theta \sim \mathcal{N}(f_{\hat{\theta}}(x), \nabla_\theta f_{\hat{\theta}}(x)^\top P^{-1} \nabla_\theta f_{\hat{\theta}}(x)), \tag{11}$$

where $\nabla_\theta f_{\hat{\theta}} \mathrel{\widehat{=}} \nabla_\theta f_\theta\Big|_{\hat{\theta}}$ is used for more compact notation. This is an instance of the Bayesian delta method (Wasserman, 2006), differing from the classic delta method by the inclusion of the prior term in $P$ in Eq. 8.

The posterior predictive depends on the form of the observation likelihood function. For regression tasks, Gaussian observation noise with fixed standard deviation $\sigma$ is the standard, resulting in posterior predictive equal to Eq. 11 plus additional variance term $\sigma^2$. For classification tasks, the posterior predictive is non-analytical, requiring Monte Carlo averaging over posterior samples. Notwithstanding the particular form of the posterior predictive distribution, the *epistemic uncertainty* of the network predictions – meaning, the error due to lack of knowledge – is represented by Eq. 11 and has considerable value in active learning, experimental design, and exploration-exploitation.

We set up the prediction-regularized joint distribution as equal to the joint distribution (Eq. 2) with an additional term proportional to the prediction at query point $x$,

$$\mathcal{L}_\theta{}^{(x,\lambda)} = \sum_{i=1}^{n} \log p(y_i \mid f_\theta(x_i)) + \log p(\theta) + \lambda f_\theta(x), \tag{12}$$

for some constant $\lambda \in \mathbb{R}$. The prediction-regularized MAP is defined as,

$$\hat{\theta}^{(x,\lambda)} \in \arg_\theta \max \mathcal{L}_\theta{}^{(x,\lambda)}. \tag{13}$$

**Theorem 1** *The derivative of the prediction under the prediction-regularized MAP w.r.t. $\lambda$ recovers the variance term in Eq. 11,*

$$\nabla_\lambda f_{\hat{\theta}^{(x,\lambda)}}(x)\Big|_{\lambda=0} = \nabla_\theta f_{\hat{\theta}}(x)^\top P^{-1} \nabla_\theta f_{\hat{\theta}}(x). \tag{14}$$

**Proof** The result is derived using the implicit delta method as detailed in Kallus and McInerney (2022). The relationship can also be understood as an application of the implicit function theorem (IFT) by Cauchy, e.g. see Lorraine et al. (2020) for a modern treatment of IFT. ■

A corollary of Theorem 1 is that detecting the local change in prediction as $\lambda$ is increased (or decreased) from 0 yields the variance term of the linearized Laplace approximation in Eq. 11. The local change, i.e. the LHS of Eq. 14, may be calculated by finite differences for some suitably small $\lambda$,

$$\frac{1}{\lambda}(f_{\hat{\theta}^{(x,\lambda)}}(x) - f_{\hat{\theta}}(x)) \underset{\lambda \to 0}{\to} \nabla_\lambda f_{\hat{\theta}^{(x,\lambda)}}(x)\Big|_{\lambda=0}. \tag{15}$$

It is natural to consider auto-differentiation to calculate the LHS of Eq. 14. From the fact that the optimum $\hat{\theta}^{(x,\lambda)}$ also changes with $\lambda$ it can be seen that auto-differentiation must deal with complex operations on optima, resulting in no computational advantage.

Putting it together, the Hessian-free Laplace (HFL) approximation is,

$$\tilde{f}_\theta \sim \mathcal{N}\left(f_{\hat{\theta}}(x), \frac{1}{\lambda}(f_{\hat{\theta}^{(x,\lambda)}}(x) - f_{\hat{\theta}}(x))\right), \tag{16}$$

and differs from linearized Laplace (Eq. 11) by the lack of an explicit Hessian. It is apparent by this comparison that HFL trades the computational and storage expense of dealing with the Hessian for that of a point-wise approximation for each query $x$. There are strong arguments in favor of the latter. First, for suitably small $\lambda$, the prediction-regularized MAP is close to the MAP for any choice of $x$, and so is not expensive to obtain with stochastic gradient descent. By comparison, matrix inversion takes $\mathcal{O}(K^3)$ computations in the number of network parameters $K$. Second, HFL sidesteps the issue of which Hessian approximation to use, a problem analogous to hyperparameter selection in that it depends on several factors including network architecture, likelihood function, and dataset.

## 3. Experiments

We explore the following data generating distributions:

- **Quadratic** Draw 32 data points $x \sim \mathcal{N}(0, 1)$ and let $y = \frac{1}{10}x^2 - \frac{1}{2}x + 5 + \frac{1}{10}\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$.

- **Sine** Fix 160 data points evenly spaced in ranges $[-1.5, -0.7)$ and $[0.35, 1.15)$ and let $y = -\sin(3x - \frac{3}{10}) + \frac{1}{10}\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$. This example closely follows Foong et al. and tests the ability to extrapolate "in-between" uncertainty (Foong et al., 2019).

A two-layer feedforward neural architecture is used with 10 hidden units used in each layer with tanh activations to ensure that the Hessian exists. For both datasets, the epistemic uncertainty of the network predicted means is plotted in 1D feature space along with the covariance matrix of predicted means in Figures 1 and 2. Four methods are compared,

- **Exact Hessian** using Eq. 8 and both terms in the likelihood Hessian (Eq. 9).

- **HFL** (this paper), using Eq. 16.

- **GGN** using Eq. 8 with only the second term in likelihood Hessian (Eq. 9).

- **Eigenvector approximation** of the GGN for $P^{-1} \approx A\Lambda^{-1}A^\top$ using the top $k$ eigenvectors $A$ of $P$ (with corresponding eigenvalues $\Lambda$) where $k = \log_e(\dim(\theta))$ rounded to the nearest positive integer. The GGN is chosen because it only makes sense to use the eigenvector approximation when the Hessian is large.

We find that, for the small-scale datasets explored, there is a noticeable difference between the exact Hessian and GGN, and this is reflected in the eigenvector approximation. This may be explained by the fact that the network clearly has not reduced the residual to near zero by fitting every data point. One would expect these characteristics to change for high dimensional features and large data sets. Interestingly, the variances from HFL are closer to the full Hessian than to GGN, running contrary to expectations given that HFL uses the linearization in Eq. 10. Finally, Figures 3 and 4 visualize the Hessians given by the the explicit Hessian methods. While the eigenvector approximation recovers some of the structure it does noticeably miss the strong diagonal variances in the parameters.

## 4. Conclusions & Future Work

In this paper we developed the technical direction toward performing the Laplace approximation without explicit Hessian evaluation and inversion (HFL). Instead, two point estimates are used, the usual MAP estimate and a regularized form of the MAP. Initial experiments show early potential of HFL in the context of Bayesian deep learning. An important step in future work is to scale up the number of parameters and neural architecture sizes along with number of features and data points to demonstrate the value of the approximation.

## References


Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *International Conference on Machine Learning*, pages 1613–1622, 2015.

Andrew YK Foong, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. 'In-between' uncertainty in Bayesian neural networks. *arXiv preprint arXiv:1906.11537*, 2019.

Alexander Immer, Maciej Korzepa, and Matthias Bauer. Improving predictions of Bayesian neural nets via local linearization. In *International Conference on Artificial Intelligence and Statistics*, pages 703–711. PMLR, 2021.

Nathan Kallus and James McInerney. The implicit delta method. *Advances in Neural Information Processing Systems*, 35:37471–37483, 2022.

Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems*, 30, 2017.

Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific Reports*, 7(1):17816, 2017.

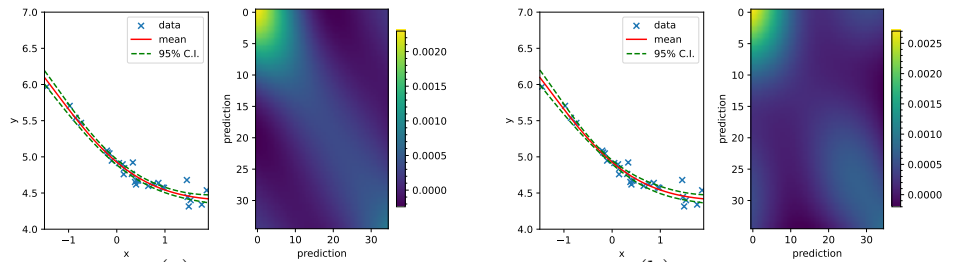
Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR, 2020.

David JC MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

David JC Mackay. Bayesian methods for adaptive models. *Doctoral Thesis, California Institute of Technology*, 1992.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable Laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, volume 6. International Conference on Representation Learning, 2018.

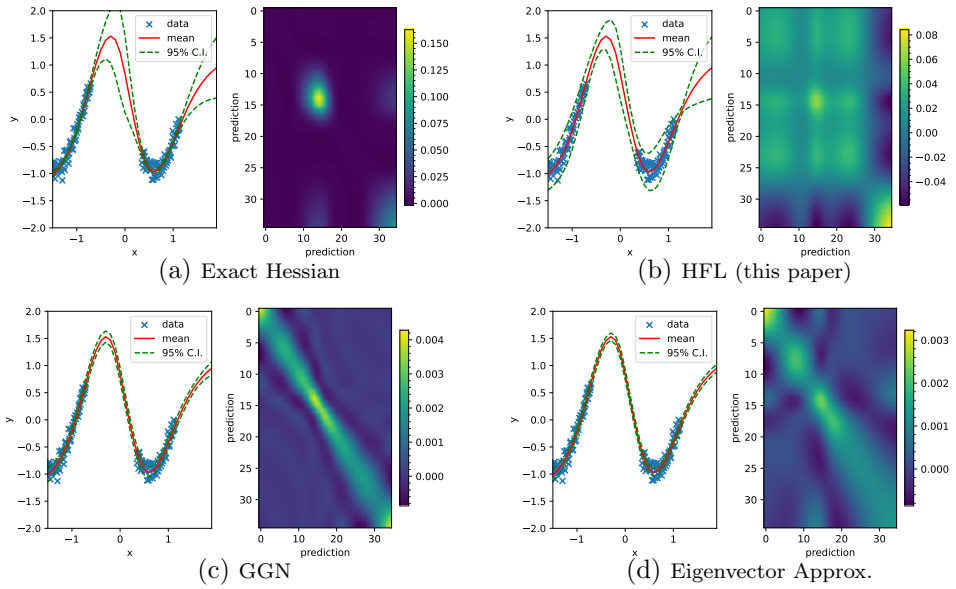
Larry Wasserman. *All of nonparametric statistics*. Springer, 2006.




(a) Exact Hessian (b) HFL (this paper)

(c) GGN (d) Eigenvector Approx.

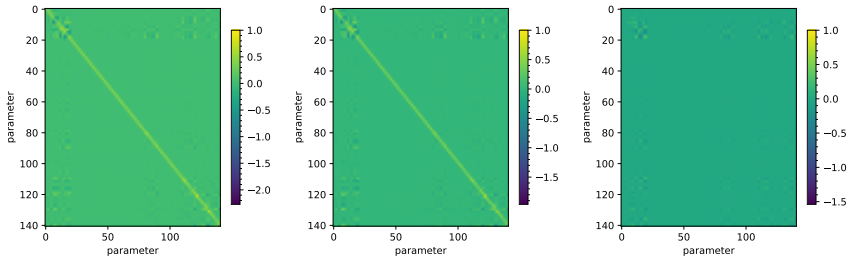
Figure 1: Fits along with uncertainty bounds and estimated prediction-covariance matrix for data generated from $y = \frac{1}{10}x^2 - \frac{1}{2}x + 5 + \frac{1}{10}\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$




(a) Exact Hessian (b) HFL (this paper)

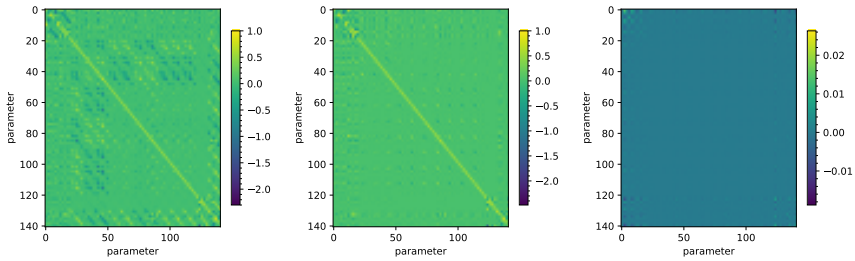
(c) GGN (d) Eigenvector Approx.

Figure 2: Fits along with uncertainty bounds and estimated prediction-covariance matrix for data generated from $y = -\sin(3x - \frac{3}{10}) + \frac{1}{10}\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$




(a) Exact Hessian (b) GGN (c) Eigenvector Approx.

Figure 3: Covariance matrix in Laplace approx. and its approximations in quadratic task.




(a) Exact Hessian (b) GGN (c) Eigenvector Approx.

Figure 4: Covariance matrix in Laplace approximation and its approximations in sine task.