Scalable Marginal Likelihood Estimation for Model Selection in Deep Learning

Alexander Immer [1 2] **Matthias Bauer** [† 3 4] **Vincent Fortuin** [1] **Gunnar Rätsch** [1 2] **Mohammad Emtiyaz Khan** [5]

Abstract

Marginal-likelihood based model-selection, even though promising, is rarely used in deep learning due to estimation difficulties. Instead, most approaches rely on validation data, which may not be readily available. In this work, we present a scalable marginal-likelihood estimation method to select both hyperparameters and network architectures, based on the training data alone. Some hyperparameters can be estimated online during training, simplifying the procedure. Our marginal-likelihood estimate is based on Laplace's method and Gauss-Newton approximations to the Hessian, and it outperforms cross-validation and manual-tuning on standard regression and image classification datasets, especially in terms of calibration and out-of-distribution detection. Our work shows that marginal likelihoods can improve generalization and be useful when validation data is unavailable (e.g., in nonstationary settings).

1. Introduction

Bayesian deep learning has made great strides on approximate inference but little has been done regarding model selection. Bayesian predictive distributions have been used to improve calibration, detect out-of-distribution data, and sometimes even improve the accuracy (Osawa et al., 2019; Maddox et al., 2019). To that end, a wide variety of scalable approximate inference methods have been explored, including Laplace's method (Ritter et al., 2018), variational approximations (Blundell et al., 2015; Khan et al., 2018), sampling methods (Wenzel et al., 2020), and ensembles (Lakshminarayanan et al., 2017). Still, there is not much work on model-selection, even though it was one of the original motivations for Bayesian neural networks (Buntine & Weigend, 1991; MacKay, 1995; Neal, 1995).

[1]Department of Computer Science, ETH Zurich, Switzerland [2]Max Planck ETH Center for Learning Systems (CLS) [3]Max Planck Institute for Intelligent Systems, Germany [4]University of Cambridge, UK [5]RIKEN Center for Advanced Intelligence Project, Japan, [†]M.B. is now at DeepMind, UK. Correspondence to: Alexander Immer <alexander.immer@inf.ethz.ch>. *Proceedings of the 38^{th} International Conference on Machine Learning*, PMLR 139, 2021.

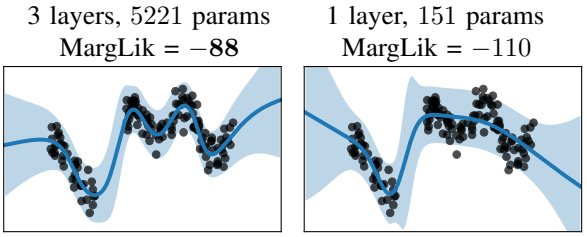

Figure 1: A deeper 3-layer network *(left)* has a better marginal likelihood compared to a 1-layer network *(right)*. This agrees with the fit where the deeper network appears to explain the 'sinusoidal' trend better. We optimize the prior and noise variance on marginal likelihood estimates *during training* (see Alg. 1 and Sec. 4.1 for details).

Bayesian model-selection uses the *marginal likelihood*—the normalizing constant of the posterior distribution—which can be estimated solely from training data and without any validation data. Also known as *empirical Bayes* (Robbins, 1955) or *type-II maximum likelihood* learning (Rasmussen & Williams, 2006), it is closely related to Occam's razor (Jefferys & Berger, 1992; Rasmussen & Ghahramani, 2001; MacKay, 2003). It can also be seen as a form of cross-validation based on the training data (Fong & Holmes, 2020), and it is commonly used in Gaussian processes (GPs) and Deep GPs to select hyperparameters (Rasmussen & Williams, 2006; Damianou & Lawrence, 2013) or even learn invariances from data (van der Wilk et al., 2018; Dutordoir et al., 2020). However, it is rarely used in deep learning because it is challenging to estimate (Llorente et al., 2020).

The marginal likelihood is intractable for even medium-sized neural-network models, and most Bayesian deep learning approaches simply use validation data (Khan et al., 2018; Zhang et al., 2018; Osawa et al., 2019). Originally, MacKay (1995) advocated the use of Laplace's method for small-scale neural networks. This requires Hessian computations and does not scale to large problems. Recently, Khan et al. (2019) showed promising results for small-scale regression using a GP viewpoint of deep networks, and Lyle et al. (2020) used infinite-width neural networks (also GPs).

The goal of this paper is to provide a scalable and online version of MacKay's original proposal, and apply it to modern, larger models used in deep learning. We propose a scalable Laplace's method to approximate the marginal like-

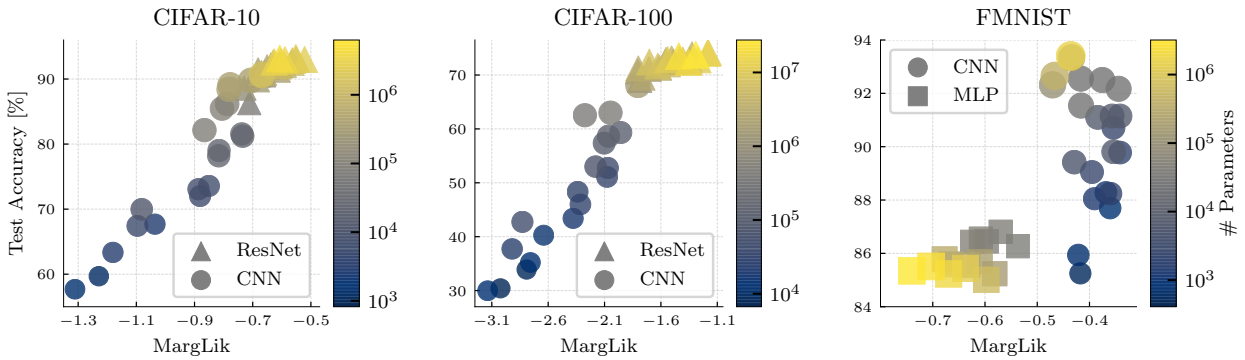

Figure 2: Each dot above shows a model of different size and/or architecture (around 40 models per plot of varying widths and depths). Models with higher training marginal-likelihood tend to have higher test accuracy. For similar performance, smaller models tend to have a higher marginal-likelihood as desired. Marker size and color changes with the number of parameters. See Sec. 4.3 for details and App. C.5 for a full list of models with accuracy and marginal-likelihood estimates.

lihood and select both the hyperparameters and network architectures. Differentiable hyperparameters can be optimized *during* training in an online fashion (unlike MacKay (1995)'s original proposal and Khan et al. (2019)'s recent approach), and discrete ones can be selected after training.

The method relies on the Generalized Gauss-Newton (GGN) and the Empirical Fisher (EF) approximation to the Hessian. The full GGN and EF can also be extremely expensive and we reduce the cost by using additional diagonal and block-diagonal approximations (Martens & Grosse, 2015; Botev et al., 2017). All these approximations have been used in approximate inference before (Khan et al., 2018; Zhang et al., 2018; Ritter et al., 2018; Osawa et al., 2019; Foresee & Hagan, 1997; Khan et al., 2019), but they have not been applied to model selection and their effectiveness for marginal likelihood estimation has also been unknown. Unlike our proposal, the method by Khan et al. (2019) is limited to regression and uses the full but intractable GGN approximation. Blundell et al. (2015) also used the variational lower-bound for hyperparameter optimization but found it to give worse results.

Our main contribution is to show that, even after making these approximations to improve scalability, the estimated marginal likelihoods can faithfully select reasonable models (see Figs. 1 and 2). Our method achieves performance on par or better than cross-validation on a range of regression and classification benchmarks. The best architecture identified by the marginal likelihood aligns with the test performance (see Fig. 2). It aligns well with the empirical observation that ResNets often perform better than the standard convolutional networks, which in turn are observed to give better results than their fully-connected counterparts. Overall, our work supports the long-held hypothesis that the marginal likelihood is effective for model selection in deep learning, and shows that a relatively cheap and simple approximation can achieve competitive results.

2. Background

In this paper, we consider supervised learning tasks with inputs $\mathbf{x}_n \in \mathbb{R}^D$ and C-dimensional vector outputs $\mathbf{y}_n$ (real or categorical outcomes), and denote the data as the set $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ of N training-example pairs.

Bayesian models. We denote by $\mathbf{f}(\mathbf{x}, \boldsymbol{\theta})$ the C-dimensional real-valued output of a neural network with parameters $\boldsymbol{\theta} \in \mathbb{R}^P$, specified by a model $\mathcal{M}$ which typically consists of a network architecture and *hyperparameters*. A Bayesian model can then be defined using a likelihood and a prior, to get the posterior distribution: $p(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M}) \propto p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M})p(\boldsymbol{\theta}|\mathcal{M})$. We assume that the data examples are sampled *i.i.d.* from $p(\mathbf{y}|\mathbf{f}(\mathbf{x}, \boldsymbol{\theta}), \mathcal{M})$. The normalizing constant of the posterior, also known as the *marginal likelihood*, is given by the following expression:

$$p(\mathcal{D}|\mathcal{M}) = \int \prod_{n=1}^N p(\mathbf{y}_n|\mathbf{f}(\mathbf{x}_n, \boldsymbol{\theta}), \mathcal{M})p(\boldsymbol{\theta}|\mathcal{M}) \, \mathrm{d}\boldsymbol{\theta}. \quad (1)$$

The model $\mathcal{M}$ might consist of the choice of network-architecture (CNN, ResNet, etc.) and hyperparameters of the likelihood and prior, for example, observation noise and prior variances. Some of these are continuous parameters while others are discrete. Our goal is to use the marginal likelihood to select such parameters.

Bayesian model comparison. The marginal likelihood in Eq. 1 can be seen as a probability distribution over the space of all datasets (of size N) given the model $\mathcal{M}$. The distribution is expected to be wider for complex models, because such models can generate data of more variety than simpler models. Given the training data $\mathcal{D}$, a model too simple or too complex will therefore be assigned a lower probability $p(\mathcal{D}|\mathcal{M})$, naturally yielding *Occam's razor* (Blumer et al., 1987; Jefferys & Berger, 1992; Rasmussen & Ghahramani, 2001). See Fig. 3.13 in Bishop (2006) for an illustration. A simple method is to pick the model that assigns the highest

probability to the training data,

$$\mathcal{M}_* = \arg\max_{\mathcal{M}} p(\mathcal{D}|\mathcal{M}). \tag{2}$$

This procedure is also called *type-II maximum likelihood* estimation or *empirical Bayes* (MacKay, 2003); it is commonly used in the Gaussian process literature to find the hyperparameters of the kernel function (Rasmussen & Williams, 2006). Additionally, we can reapply Bayes rule with the marginal likelihood as the *likelihood* and an additional prior distribution $p(\mathcal{M})$ over the models (MacKay, 1992; 2003). Eq. 2 can then be seen as a special case of a *maximum a-posteriori* (MAP) estimate with a uniform prior.

Laplace's method. Using the marginal likelihood for neural-network model selection was originally proposed by MacKay (1992), who used Laplace's method to approximate Eq. 1. The method relies on a local quadratic approximation of $\log p(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M})$, around a maximum $\boldsymbol{\theta}_*$, resulting in a Gaussian approximation to $p(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M})$, denoted by $q(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M})$, and an approximation to the marginal likelihood, denoted by $q(\mathcal{D}|\mathcal{M})$ and shown below,

$$\begin{aligned}\log p(\mathcal{D}|\mathcal{M}) &\approx \log q(\mathcal{D}|\mathcal{M}) \\ &:= \log p(\mathcal{D}, \boldsymbol{\theta}_*|\mathcal{M}) - \tfrac{1}{2}\log\left|\tfrac{1}{2\pi}\mathbf{H}_{\boldsymbol{\theta}_*}\right|,\end{aligned} \tag{3}$$

with Hessian $\mathbf{H}_{\boldsymbol{\theta}} := -\nabla^2_{\boldsymbol{\theta}\boldsymbol{\theta}} \log p(\mathcal{D}, \boldsymbol{\theta}|\mathcal{M})$. However, computing $\mathbf{H}_{\boldsymbol{\theta}}$, a large matrix of size $P \times P$, and its determinant is infeasible in general. We refer to the log marginal likelihood as *margLik* and report its value normalized by the number of training examples.

Using the above objective to select models can also lead to better generalization. The log-determinant in Eq. 3 favors Hessians with small eigenvalues and is thus closely related to low curvature and flatness of a solution's neighborhood. Such *flat minima* have been argued to generalize better than sharper ones (Hochreiter & Schmidhuber, 1997), a claim that is supported empirically (Keskar et al., 2016; Jiang et al., 2019; Maddox et al., 2019) and has also been formalized using PAC-Bayesian generalization bounds (Dziugaite & Roy, 2017). We can therefore expect models selected using the marginal-likelihood to have similar properties. The local Gaussian approximation above may not always be sufficient, and the goal of this paper is to investigate its capacity to measure complexity of large neural networks.

Hessian approximations. Recently, scalable approximations to the Hessian $\mathbf{H}_{\boldsymbol{\theta}}$, such as those based on the generalized Gauss-Newton (GGN) and empirical Fisher (EF), have been applied to approximate Bayesian inference (Khan et al., 2018; Zhang et al., 2018; Ritter et al., 2018; Osawa et al., 2019; Immer et al., 2021; Kristiadi et al., 2020), but their application to marginal-likelihood estimation or model selection has not yet been explored, except by Khan et al. (2019) on small regression tasks with a full GGN.

The GGN approximation to the log-joint Hessian is based on the *Jacobian* matrix $\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x}) \in \mathbb{R}^{C\times P}$ of the network features with entries $[\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x})]_{ci} = \frac{\partial f_c(\mathbf{x},\boldsymbol{\theta})}{\partial \theta_i}$, as well as the Hessians of the log-likelihood and the log-prior,

$$\boldsymbol{\Lambda}(\mathbf{y}; \mathbf{f}) := -\nabla^2_{\mathbf{f}\mathbf{f}} \log p(\mathbf{y}|\mathbf{f}), \quad \mathbf{P}_{\boldsymbol{\theta}} := -\nabla^2_{\boldsymbol{\theta}\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\mathcal{M}),$$

respectively. The GGN approximation is then given by

$$\mathbf{H}_{\boldsymbol{\theta}} \approx \mathbf{H}^{\text{GGN}}_{\boldsymbol{\theta}} = \mathbf{J}_{\boldsymbol{\theta}}^\intercal \mathbf{L}_{\boldsymbol{\theta}} \mathbf{J}_{\boldsymbol{\theta}} + \mathbf{P}_{\boldsymbol{\theta}}, \tag{4}$$

where $\mathbf{J}_{\boldsymbol{\theta}}$ is an $NC \times P$ matrix formed by *stacking* N Jacobians $\mathbf{J}_{\boldsymbol{\theta}}(\mathbf{x}_n)$ overall $\mathbf{x}_n$ in $\mathcal{D}$, and $\mathbf{L}_{\boldsymbol{\theta}}$ is a $NC \times NC$ block-diagonal matrix with blocks $\boldsymbol{\Lambda}(\mathbf{y}_n; \mathbf{f}(\mathbf{x}_n, \boldsymbol{\theta}))$.

The expression for the GGN approximation in Eq. 4 is equivalent to the Fisher information matrix (Kunstner et al., 2019); we therefore also consider the *empirical Fisher* (EF) approximation to the Hessian, which relies on an outer product of the gradients $\mathbf{G}_{\boldsymbol{\theta}}(\mathbf{x}) = \nabla_{\boldsymbol{\theta}} \log p(\mathbf{y}|\mathbf{f}(\mathbf{x}, \boldsymbol{\theta}), \mathcal{M}) \in \mathbb{R}^P$:

$$\mathbf{H}_{\boldsymbol{\theta}} \approx \mathbf{H}^{\text{EF}}_{\boldsymbol{\theta}} = \mathbf{G}_{\boldsymbol{\theta}}^\intercal \mathbf{G}_{\boldsymbol{\theta}} + \mathbf{P}_{\boldsymbol{\theta}}, \tag{5}$$

where $\mathbf{G}_{\boldsymbol{\theta}}$ is an $N \times P$ matrix of stacked gradients $\mathbf{G}_{\boldsymbol{\theta}}(\mathbf{x}_n)$ for all $\mathbf{x}_n \in \mathcal{D}$. The complexities of the GGN and EF are $\mathcal{O}(P^2NC + PNC^2)$ and $\mathcal{O}(P^2N)$, respectively. The EF is therefore $\mathcal{O}(C)$ cheaper to compute as C is typically much smaller than P. For details, see App. B. Throughout, we use "Laplace-GGN" and "Laplace-EF" to refer to Laplace's method where the Hessian has been approximated by using the GGN or EF, respectively.

3. Model Selection with Laplace-GGN and -EF

We propose to use GGN and EF approximations for scalable marginal-likelihood estimation based on Laplace's method. Our method involves two key steps:

Step 1. During training, we update differentiable hyperparameters of $\mathcal{M}$ by using gradients of Eq. 3. See line 8 in Alg. 1 and cf. Fig. 3 *(left)* for an example.

Step 2. After training, we use the marginal likelihood of the trained model to make discrete choices, e.g., to select the network architecture as shown in Fig. 1, or to choose between several trained models, shown in Fig. 3 *(right)*.

We first describe each of these steps and then give details about efficient Hessian-determinant computations.

3.1. Step 1: Online model selection during training

Here we alternate between optimizing the network parameters $\boldsymbol{\theta}$ and updating the continuous hyperparameters $\mathcal{M}^\partial \subseteq \mathcal{M}$ that appear in the marginal likelihood in a differentiable way. Such differentiable hyperparameters may include the observation noise of the likelihood, the variances of the Gaussian prior, or the softmax temperature parameter. The resulting algorithm is outlined in Alg. 1.

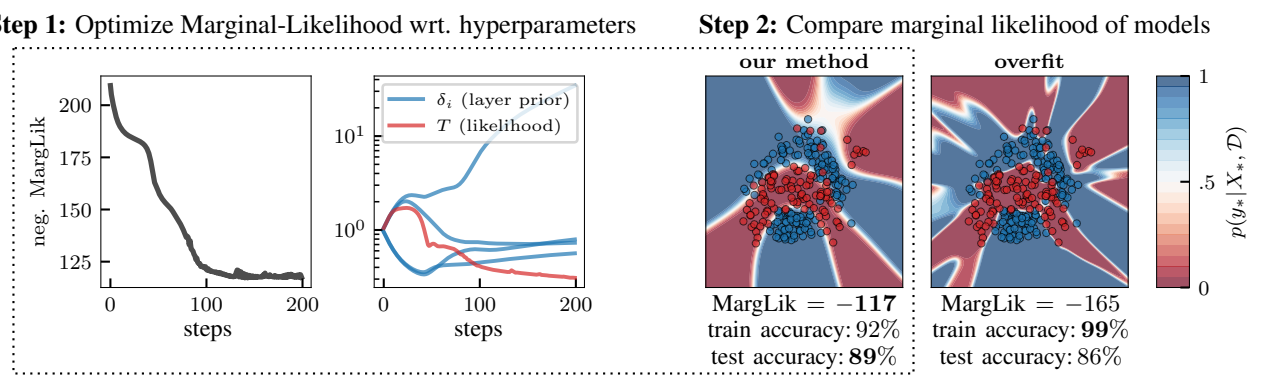

Figure 3: Proposed method for model selection using the marginal likelihood. In Step 1, we apply our online algorithm (Alg. 1) to optimize the marginal likelihood estimate (Eq. 3) with respect to the differentiable hyperparameters (*here*: prior precision δ_i per layer and softmax temperature T). In Step 2, we compare the resulting model (*left*) to an overfitting model (*right*) with higher training accuracy but lower test accuracy; both models have the same architecture. The Laplace-GGN marginal likelihood estimate $\log q(\mathcal{D}|\mathcal{M})$ correctly identifies the model that generalizes better. See Sec. 4.1 for details.

To optimize the network parameters $\boldsymbol{\theta}$ we perform regular neural network training on the *maximum a posteriori* (MAP) objective in Eq. 6 using stochastic optimizers like ADAM (Kingma & Ba, 2015) or SGD (cf. line 4 in Alg. 1),

$$\log p(\mathcal{D}, \boldsymbol{\theta}|\mathcal{M}) = \sum_{n=1}^{N} \log p(\mathbf{y}_n|\mathbf{f}(\mathbf{x}_n, \boldsymbol{\theta})) + \log p(\boldsymbol{\theta}). \quad (6)$$

To optimize the continuous hyperparameters $\mathcal{M}^\partial$, we perform gradient ascent on the marginal likelihood estimate (Eq. 3). Because of computational considerations further detailed in Sec. 3.4, we only do so every F epochs after an initial burn-in period of B epochs and then perform K update steps with step size $\gamma > 0$ (cf. lines 6-10 in Alg. 1):

$$\mathcal{M}^\partial \leftarrow \mathcal{M}^\partial + \gamma \nabla_{\mathcal{M}^\partial} \log q(\mathcal{D}|\mathcal{M}). \quad (7)$$

The marginal likelihood estimate can be updated cheaply for each of the K iterations (cf. App. B). Fig. 3 *(left)* illustrates this optimization of the Gaussian prior-variances for each layer as well as the softmax temperature of the likelihood. We can also use the marginal-likelihood for early stopping *without* validation data.

Note that the parameters $\boldsymbol{\theta}_*$ in Eq. 3 are assumed to be the MAP estimate, however this is not true during training at some $\boldsymbol{\theta}$. We also try another method derived from a local integration in App. A.1 instead, but empirically this does not yield good results and is more expensive. We discuss the choice of hyperparameters of Alg. 1 in Sec. 3.4.

3.2. Step 2: Model selection after training

To choose between two discrete model alternatives, such as different architectures, we compare their marginal likelihood estimates after training. This step is a basic hypothesis test, where we compare two models $\mathcal{M}$ and $\mathcal{M}'$ and choose the more likely model given the data according to the likelihood ratio $p(\mathcal{D}|\mathcal{M})/p(\mathcal{D}|\mathcal{M}')$, which is the most powerful statistical test for this purpose (Neyman & Pearson, 1933). In terms of the marginal likelihood, we only need to choose the model with a higher value (cf. Fig. 1 and Fig. 3 *(right)*).

Algorithm 1 Marginal likelihood based training

1: **Input:** dataset $\mathcal{D}$, likelihood $p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M})$, prior $p(\boldsymbol{\theta}|\mathcal{M})$. Initial model $\mathcal{M}$, step size γ, steps K, burn-in epochs B, marglik frequency F. GGN or EF.
2: initialize $\boldsymbol{\theta}$ of the neural network
3: **for** each *epoch* **do**
4: $\quad\boldsymbol{\theta} \leftarrow trainEpoch$(objective in Eq. 6)
5: $\quad$**if** $epoch > B$ and $epoch \mod F = 0$ **then**
6: $\quad\quad$compute $\log q(\mathcal{D}|\mathcal{M})$ (Eq. 3) with $\mathbf{H}^{\text{GGN}}_{\boldsymbol{\theta}}$ or $\mathbf{H}^{\text{EF}}_{\boldsymbol{\theta}}$
7: $\quad\quad$**for** K steps **do**
8: $\quad\quad\quad\mathcal{M}^\partial \leftarrow \mathcal{M}^\partial + \gamma \nabla_{\mathcal{M}^\partial} \log q(\mathcal{D}|\mathcal{M})$
9: $\quad\quad\quad$update $\mathcal{M}$ and $\log q(\mathcal{D}|\mathcal{M})$
10: $\quad\quad$**end for**
11: $\quad$**end if**
12: **end for**
13: Return marginal likelihood $\log q(\mathcal{D}|\mathcal{M})$ and optionally posterior approximation $q(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M})$.

3.3. Scalable Laplace approximations

Efficient determinant computation. Scalable marginal likelihood estimation (Eq. 3) relies on an efficient computation of the determinant of the GGN or EF approximation of the Hessian (Eqs. 4 and 5). When N is small, we can use the Woodbury matrix identity to rewrite the determinant of the Hessian (a $P \times P$ matrix) in terms of determinants of matrices whose size only depends on the number of data

points N and outputs (e.g., classes) C instead:

$$|\mathbf{H}^{\text{GGN}}_{\boldsymbol{\theta}}| = |\underbrace{\mathbf{J}_{\boldsymbol{\theta}}^\intercal \mathbf{L}_{\boldsymbol{\theta}} \mathbf{J}_{\boldsymbol{\theta}} + \mathbf{P}_{\boldsymbol{\theta}}}_{P \times P}| = |\underbrace{\mathbf{J}_{\boldsymbol{\theta}} \mathbf{P}_{\boldsymbol{\theta}}^{-1} \mathbf{J}_{\boldsymbol{\theta}}^\intercal + \mathbf{L}_{\boldsymbol{\theta}}^{-1}}_{NC \times NC}||\mathbf{L}_{\boldsymbol{\theta}}||\mathbf{P}_{\boldsymbol{\theta}}|$$
$$|\mathbf{H}^{\text{EF}}_{\boldsymbol{\theta}}| = |\underbrace{\mathbf{G}_{\boldsymbol{\theta}}^\intercal \mathbf{G}_{\boldsymbol{\theta}} + \mathbf{P}_{\boldsymbol{\theta}}}_{P \times P}| = |\underbrace{\mathbf{G}_{\boldsymbol{\theta}} \mathbf{P}_{\boldsymbol{\theta}}^{-1} \mathbf{G}_{\boldsymbol{\theta}}^\intercal + \mathbf{I}_N}_{N \times N}||\mathbf{P}_{\boldsymbol{\theta}}|. \quad (8)$$

The determinants $|\mathbf{P}_{\boldsymbol{\theta}}|$ (though still $P \times P$) and $|\mathbf{L}_{\boldsymbol{\theta}}|$ are usually cheap to compute as the prior $p(\boldsymbol{\theta})$ often factorizes across parameters and $\mathbf{L}_{\boldsymbol{\theta}}$ is block-diagonal. When neither $\mathcal{O}(N^3)$ nor $\mathcal{O}(P^3)$ are tractable, we consider the following structured GGN approximations of different sparsities.

Kronecker-factored Laplace. The Kronecker-factored (KFAC) GGN approximation is based on a block-diagonal approximation to $\mathbf{H}^{\text{GGN}}_{\boldsymbol{\theta}}$ and is specified by a Kronecker product per layer (Martens & Grosse, 2015; Botev et al., 2017). The GGN of the l-th layer of the neural network is approximated as $[\mathbf{J}_{\boldsymbol{\theta}}^\intercal \mathbf{L}_{\boldsymbol{\theta}} \mathbf{J}_{\boldsymbol{\theta}}]_l \approx \mathbf{Q}_l \otimes \mathbf{W}_l$ where $\mathbf{Q}_l$ is computed from the gradient by backpropagation and $\mathbf{W}_l$ depends on the input to the l-th layer. $\mathbf{W}_l$ and $\mathbf{Q}_l$ are quadratic in the l-th layer's input and output size, respectively. Let $\mathbf{q}^{(l)} \in \mathbb{R}^{D_l}$ and $\mathbf{w}^{(l)} \in \mathbb{R}^{D'_l}$ be the eigenvalues of $\mathbf{Q}_l$ and $\mathbf{W}_l$, respectively. If the prior Hessian $\mathbf{P}_{\boldsymbol{\theta}}$ is isotropic per layer, that is, the l-th block-diagonal entry of $\mathbf{P}_{\boldsymbol{\theta}}$ is given by $p_\theta^{(l)} \mathbf{I}_l$, then we can compute the determinant for the Laplace-GGN efficiently as

$$|\mathbf{H}^{\text{GGN}}_{\boldsymbol{\theta}}| \approx |\mathbf{H}^{\text{KFAC}}_{\boldsymbol{\theta}}| = \prod_l \prod_{ij} \mathbf{q}_i^{(l)} \mathbf{w}_j^{(l)} + p_\theta^{(l)}. \quad (9)$$

In contrast to the typical use of Kronecker-factored approximations in optimization (Martens & Grosse, 2015; Botev et al., 2017) and approximate inference (Ritter et al., 2018; Zhang et al., 2018), we do not use damping which would distort the Laplace-GGN (cf. App. A.3 for discussion and ablation experiment). Computationally, the Kronecker-factored Laplace-GGN is cheaper than the full Laplace-GGN because we only need to decompose matrices that are quadratic in the number of neurons per layer. This number typically does not exceed a few thousand.

Diagonal Laplace. The diagonal Laplace approximates the GGN or EF by their diagonal, which allows for the cheap computation of the determinants

$$|\mathbf{H}^{\text{GGN}}_{\boldsymbol{\theta}}| \approx \prod_p [\mathbf{H}^{\text{GGN}}_{\boldsymbol{\theta}}]_{pp} \text{ and } |\mathbf{H}^{\text{EF}}_{\boldsymbol{\theta}}| \approx \prod_p [\mathbf{H}^{\text{EF}}_{\boldsymbol{\theta}}]_{pp}. \quad (10)$$

Here, $[\mathbf{H}]_{pp}$ denotes the diagonal entries of $\mathbf{H}$. The computation of the diagonal GGN still scales with the number of outputs C due to the dependence on the block-diagonal $\mathbf{L}_{\boldsymbol{\theta}}$. In contrast, the diagonal EF is very cheap to compute because it only requires the sum of point-wise squared individual gradients over the training data. Despite their simplicity and the hypothesis by MacKay (1992) that they are "no good", we find that these approximations work well in practice. Especially for convolutional neural networks, the diagonal approximations provide a cost-effective alternative to block-diagonal approximations when used in Alg. 1. This is in contrast to the predictive performance of diagonal Laplace approximations, which are often significantly worse their less sparse counterparts (Ritter et al., 2018; Immer et al., 2021).

3.4. Computational considerations

The overall runtime of our algorithm depends on the frequency F with which we estimate the marginal likelihood, the number of consecutive hyperparameter update steps K, as well as the burn-in period B. Furthermore, the cost of each marginal likelihood estimation (line 6 in Alg. 1) and marginal likelihood update (line 9 in Alg. 1) depends on the specific approximation of the Hessian used. Here, we discuss the most important aspects briefly. See App. B for more details and discussion, including computation and memory complexities.

Estimation of the marginal likelihood (Eq. 3) requires one full pass over the training data: While its first term, the log joint $p(\mathcal{D}, \boldsymbol{\theta}|\mathcal{M})$ (Eq. 6), decomposes into individual terms per data point and can, therefore, be estimated on minibatches, this is not the case for the log determinant of the Hessian. However, this initial estimation cost amortizes over the K consecutive update steps for the hyperparameters (Alg. 1 lines 7 to 10), as the most expensive computations can be reused. Therefore, it makes sense to perform many update steps (K large) but do so at a lower frequency (estimation frequency F low). We typically use $K = 100$ update steps in large-scale experiments and perform marginal likelihood estimation every $F = 1$ epochs for small-scale experiments or $F = 5$ to 10 epochs for larger experiments. In contrast to MacKay (1995), these updates are continuous during training, instead of a single update and re-training which would be prohibitive.

We use ADAM for the gradient updates of the hyperparameters in Eq. 7 and line 8 of Alg. 1. We can use a slightly higher learning rate than default (γ between 0.01 and 1) without divergence because the gradients are not stochastic (Bottou, 2010). Similar to the approach by Lyle et al. (2020), our marginal likelihood estimate can be used for early stopping or to save models so that we end up using the model that achieved the best marginal likelihood during training in Alg. 1. Overall, we find that the online algorithm is robust to different settings of K, F, B, and γ, and it is not difficult to monitor convergence of the marginal likelihood and hyperparameters, of which there are fewer than model parameters. In our experiments, we reliably optimize up to around 400 hyperparameters in the case of a ResNet. Typically, our algorithm does not need more epochs to converge than standard training. For further discussion of the robustness to hyperparameters, see App. C.6.

4. Experiments

We compare our method to model selection by cross-validation and also to manually selected models. Depending on the problem size, we use different variants of the Laplace-GGN approximation. On smaller scale UCI (Dua & Graff, 2017) and toy examples, we use the full GGN and EF determinants and the Kronecker-factored approximation. On larger problems, we use the Kronecker-factored and the diagonal Laplace-GGN and EF approximations.

We also compare the posterior predictive obtained by a Laplace-GGN posterior approximation (Ritter et al., 2018; Immer et al., 2021), defined in Eq. 11, to the MAP prediction, defined in Eq. 12,

$$p_{\text{dist}}(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = \frac{1}{S}\sum_s^S p(\mathbf{y}^*|\mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}_*}(\mathbf{x}^*, \boldsymbol{\theta}_s), \mathcal{M}_*), \quad (11)$$

$$p_{\text{MAP}}(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = p(\mathbf{y}^*|\mathbf{f}(\mathbf{x}^*, \boldsymbol{\theta}_*), \mathcal{M}_*). \quad (12)$$

Here, $\mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}_*}(\mathbf{x}, \boldsymbol{\theta}_s) = \mathbf{f}(\mathbf{x}, \boldsymbol{\theta}_*) + \mathbf{J}_{\boldsymbol{\theta}_*}(\mathbf{x})(\boldsymbol{\theta} - \boldsymbol{\theta}_*)$ is the linearized neural network at $\boldsymbol{\theta}_*$, and $\boldsymbol{\theta}_s \sim q(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M}_*)$.

In all experiments, we optimize the precision parameter $\delta_l > 0$ of a Gaussian prior $\mathcal{N}(0, \delta_l^{-1}\mathbf{I}_l)$ for the weights and biases of each layer l individually and initialize all δ to 1. In regression experiments, we additionally optimize the Gaussian observation noise. We use ADAM (Kingma & Ba, 2015) to optimize the hyperparameters during training with default settings but a higher learning rate between 0.01 and 1 (cf. Sec. 3.4). We optimize the network parameters using ADAM except for the ResNet experiments, where we follow common practice and use SGD with momentum of 0.9. On the image classification datasets, train for 300 epochs in total and decay the learning rate by a factor of 0.1 (He et al., 2016) after 150, 225, and 275 epochs, respectively.

4.1. Illustrative examples

First, we illustrate our model selection approach on simple regression (Fig. 1) and classification (Fig. 3) examples.

Regression. For the regression example we consider the modified Snelson dataset (Snelson, 2007; Khan et al., 2019), see Fig. 1. We run Alg. 1 for two different architectures (1 and 3 hidden layers, 50 neurons per layer) with a step size of 0.01 for parameters and hyperparameters and recompute the marginal likelihood after each epoch ($F = 1$) and with no burn-in ($B = 0$) for 1000 epochs with $K = 1$ hyperparameter updates per step. We use the Laplace-GGN with a full GGN determinant and compute the Bayesian predictive in Eq. 11 based on the optional Laplace-GGN approximation.

The larger network ($P = 5221$ parameters) fits the data well and does not overfit due to the optimized regularization (Fig. 1 *(left)*). In contrast, the smaller network ($P = 151$ parameters) underfits the data due to its limited capacity and explains parts of the sinusoidal signal with a higher predictive uncertainty (Fig. 1 *(right)*). The marginal likelihood identifies the better model. In App. C.1, we show in addition how the resulting Bayesian predictive decomposes into *epistemic* and *aleatoric* uncertainty. Such a decomposition is only possible with optimized hyperparameters and would require a large grid-search without our online method (Khan et al., 2019).

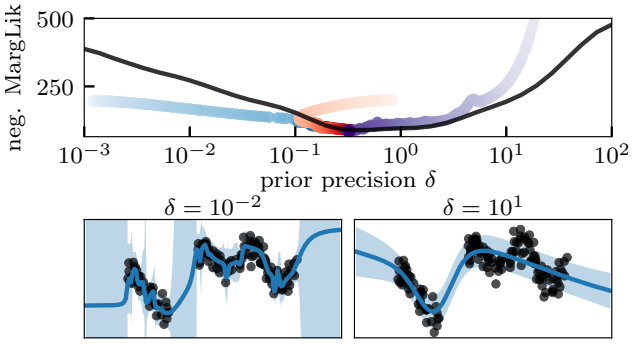

Figure 4: Selection of the prior precision δ for a 3-layer network using the marginal likelihood. *Top:* Three optimization traces (init final) of the online algorithm compared to a grid-search with fixed δ (—). The online algorithm is initialized with three different prior precisions $\delta_{\text{init}} \in \{10^{-3}, 1, 10^{2}\}$ and converges to the optimum as found by the grid-search within the same number of steps. *Bottom:* Predictive distributions of an over- and underfitting model, respectively. The predictive of the best model is depicted in Fig. 1 *(left)*. The marginal likelihood correctly identifies the model that agrees with intuition.

The larger model has enough capacity to overfit on this dataset. To illustrate the marginal likelihood's ability to select the right amount of regularization and to test the stability of our algorithm, we perform a grid search over the prior precision for the larger network and compare the selected hyperparameters. The online algorithm reliably converges to the same optimum also identified by comparing the marginal likelihood over the grid search after training (optimization traces in Fig. 4 *(top)*). Moreover, the marginal likelihood correctly identifies the value with the best generalization performance (test log likelihood in App. C.1); choosing hyperparameter values that correspond to worse marginal likelihoods results in overfitting or underfitting respectively, see the predictive distributions in Fig. 4 *(bottom)* and compare them to the optimal solution in Fig. 1 *(left)*.

Classification. For binary classification, we consider a subsampled ($N = 265$ data points) version of the synthetic 2D banana dataset (Rätsch et al., 2001). We train the same neural network (1 hidden layer with 50 neurons) twice: once with our online algorithm for 200 steps with otherwise same settings as for the regression example above; and a second time with a weakly regularized prior akin to regular deep learning training. The optimization trace and the evolution

Dataset	cross-validation (CV)			MargLik optimization		
	MAP	VI	Laplace	GGN	EF	KFAC
boston	$2.71_{\pm 0.10}$	$\mathbf{2.58}_{\pm 0.03}$	$\mathbf{2.57}_{\pm 0.05}$	$2.69_{\pm 0.12}$	$\mathbf{2.62}_{\pm 0.12}$	$2.70_{\pm 0.09}$
concrete	$3.17_{\pm 0.04}$	$3.17_{\pm 0.01}$	$\mathbf{3.05}_{\pm 0.04}$	$\mathbf{3.06}_{\pm 0.05}$	$\mathbf{3.07}_{\pm 0.04}$	$3.13_{\pm 0.06}$
energy	$1.02_{\pm 0.05}$	$1.42_{\pm 0.01}$	$0.82_{\pm 0.03}$	$\mathbf{0.55}_{\pm 0.11}$	$0.73_{\pm 0.15}$	$\mathbf{0.53}_{\pm 0.02}$
kin8nm	$-1.09_{\pm 0.01}$	$-0.87_{\pm 0.00}$	$\mathbf{-1.23}_{\pm 0.01}$	$-1.14_{\pm 0.01}$	$-1.14_{\pm 0.01}$	$-1.13_{\pm 0.01}$
naval	$-5.75_{\pm 0.05}$	$-3.82_{\pm 0.02}$	$-6.40_{\pm 0.06}$	$\mathbf{-6.93}_{\pm 0.03}$	$\mathbf{-6.92}_{\pm 0.04}$	$-6.88_{\pm 0.05}$
power	$2.82_{\pm 0.01}$	$2.86_{\pm 0.01}$	$2.83_{\pm 0.01}$	$\mathbf{2.78}_{\pm 0.02}$	$\mathbf{2.78}_{\pm 0.01}$	$\mathbf{2.78}_{\pm 0.02}$
wine	$0.98_{\pm 0.02}$	$0.96_{\pm 0.01}$	$0.97_{\pm 0.02}$	$\mathbf{0.93}_{\pm 0.02}$	$\mathbf{0.93}_{\pm 0.01}$	$\mathbf{0.94}_{\pm 0.02}$
yacht	$2.30_{\pm 0.02}$	$1.67_{\pm 0.01}$	$\mathbf{1.01}_{\pm 0.05}$	$5.89_{\pm 1.25}$	$2.43_{\pm 0.61}$	$1.48_{\pm 0.07}$

Table 1: Negative test log likelihood (lower is better) on the UCI regression benchmark. Three leftmost columns: cross-validation (CV) using MAP, VI and Laplace predictives; three rightmost columns: our proposed method with GGN, EF, and GGN-KFAC approximation to the marginal likelihood and MAP predictive. Even compared to VI and Laplace with CV (second and third column), the MAP predictions obtained with our method perform on par or better except on the kin8nm and yacht datasets. The full GGN approximation performs the best, followed by the EF and Kronecker Laplace-GGN approximations. The cross-validation results are taken from Foong et al. (2019). Results within one standard error of the best performance are in bold. Additional results with diagonal GGN and EF as well as Bayesian predictive are in App. C.

of the hyperparameters (softmax temperature as well as prior precision per layer) for our online algorithm is shown in Fig. 3 *(left)*.

The weakly regularized model overfits and has a large generalization gap with a very high accuracy of almost 100% on the train set but only 86% on the test set (Fig. 3, Step 2 *(right)*). In contrast, our online method reduces this gap and increases the test accuracy to 89% by optimizing the hyperparameters during training using the marginal likelihood (Fig. 3, Step 2 *(left)*). The optional Bayesian predictive also profits from the optimized hyperparameters (Fig. C.1b). A comparison of the marginal likelihood estimates after training correctly identifies the model with smaller generalization gap and better test performance.

4.2. UCI regression

We compare our method to cross-validation on eight UCI (Dua & Graff, 2017) regression datasets following Hernández-Lobato & Adams (2015) and Foong et al. (2019). In this setup, each dataset is split into 90% training and 10% testing data and a neural network with a single hidden layer, 50 neurons, and a `ReLU` activation is used. We use the hand-tuned and cross-validated results reported by Foong et al. (2019) who compare MAP, variational inference, and the Laplace approximation. They use 10% of the training data for validation and optimize hyperparameters by grid-search. We run Alg. 1 for $10,000$ epochs until convergence with the standard learning rate of ADAM for both hyperparameters and parameters, and set frequency $F = 1$, $K = 1$ hyperparameter gradient steps, and do not use burn-in. We compute standard errors over 10 random splits.

The results in Table 1 indicate that our online algorithm overall performs better than cross-validated baselines in terms of the test log likelihood. Using the simple and efficient MAP predictive, our method outperforms or matches the Bayesian predictions with cross-validated hyperparameters on 6 out of 8 datasets. Our algorithm performs similarly using the full GGN or EF, and the Kronecker-factored GGN performs almost as good as the full GGN. See App. C.2 and App. C.3 for further details and results on UCI classification datasets, including diagonal GGN and EF approximations. Perhaps surprisingly, the diagonal approximations are effective and also competitive with cross-validation. We further find that our method can alleviate the need for the posterior predictive as it achieves the same performance using only the MAP predictive (performance comparison in App. C.2).

4.3. Image classification

We benchmark our online model selection against cross-validation on standard image classification datasets (MNIST, FMNIST, CIFAR-10, CIFAR-100) and use the resulting marginal likelihood estimate to compare architectures. We compare fully-connected (MLP), convolutional (CNN), and residual (ResNet) networks. Our algorithm is robust to the selection of its hyperparameters (F, K, B, γ, see Sec. 3 and App. C.6). Here, we use our online model selection step every $F = 10$ epochs for $K = 100$ hyperparameter steps without burn-in and with step size $\gamma = 1$ to keep computational overhead low. Due to the size of the datasets and models, we use the Kronecker-factored GGN approximation and the diagonal EF and GGN variants. For the baseline, we use the results and architectures by Immer et al. (2021) who optimized the prior precision using cross-validation, except for the ResNet on CIFAR-10 where we use the hyperparameters of Zhang et al. (2019b). For ResNets, we use fixup (Zhang et al., 2019b) instead of batchnorm because batchnorm conflicts with a Gaussian prior (Zhang et al., 2019a). To estimate standard errors, we run for 5 different random initializations.

		cross-validation		marginal likelihood optimization					
				KFAC			diagonal EF		
Dataset	**Model**	**accuracy**	**logLik**	**accuracy**	**logLik**	**MargLik**	**accuracy**	**logLik**	**MargLik**
MNIST	**MLP**	98.22	−0.061	98.38	−0.053	−0.158	97.05	−0.095	−0.553
	CNN	99.40	**−0.017**	**99.46**	**−0.016**	**−0.064**	**99.45**	−0.019	−0.134
FMNIST	**MLP**	88.09	−0.347	89.83	−0.305	−0.468	85.72	−0.400	−0.756
	CNN	91.39	−0.258	**92.06**	**−0.233**	**−0.401**	91.69	**−0.233**	−0.570
CIFAR10	**CNN**	77.41	−0.680	80.46	−0.644	−0.967	80.17	−0.600	−1.359
	ResNet	83.73	−1.060	**86.11**	−0.595	**−0.717**	**85.82**	**−0.464**	−0.876

Table 2: Model selection by marginal likelihood compared with cross-validation on image classification. We report the test accuracy, test log likelihood (logLik), and log marginal likelihood (MargLik). Higher is better for all metrics. Generally, models selected by our method perform better than the cross-validated models, in particular on CIFAR-10. More importantly, higher accuracies correspond to higher marginal-likelihood. The marginal likelihood agrees with the intuition that CNNs are better than MLPs and ResNets are better than CNNs. Performance reported over 5 random initializations, best performance per dataset within one standard error in bold. Cross-validation results on CNN and MLP are taken from Immer et al. (2021). Results with standard errors and performance of diagonal GGN are reported in App. C.4.

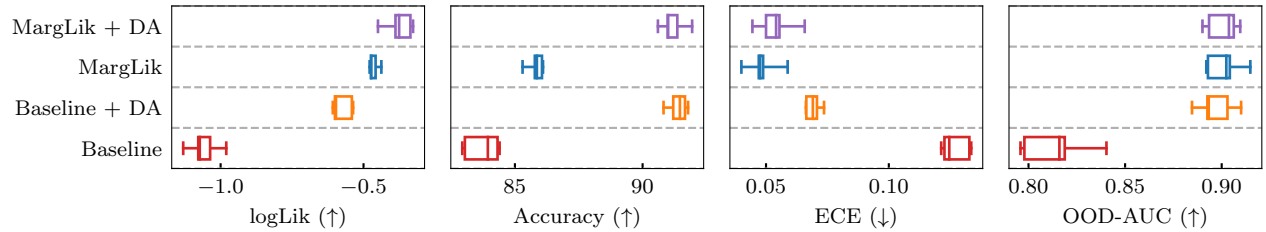

Figure 5: Comparison to a baseline on the ResNet-20 architecture with and without data augmentation (*DA*) on CIFAR-10. We report test log-likelihood (logLik), test accuracy, expected calibration error (ECE), and out-of-distribution (OOD) detection performance in terms of area under the ROC (*OOD-AUC*) using the SVHN validation set as OOD data. Arrows indicate if higher (↑) or lower (↓) values are better. Even without data augmentation, our MargLik-based method achieves a logLik, ECE, and OOD-AUC better than the baseline with data augmentation. Our approach paired with data augmentation slightly improves on that and matches the accuracy obtained by the baseline with data augmentation.

The results in Table 2 show that our method improves the performance over baselines on the considered architectures with the largest improvement on CIFAR-10. We find that the Kronecker-factored version is well-suited for fully-connected networks but the diagonal variants are often sufficient, on CIFAR-10 even slightly better, for convolutional neural networks. The resulting marginal likelihood estimates of any method suggest that CNNs are better suited for image classification than MLPs, and on CIFAR-10 we further see that ResNets are superior to standard CNNs without residual connections. In App. C.4, we additionally show the outcome with the diagonal GGN and list standard errors.

In Fig. 5, we show results with additional data augmentation on CIFAR-10 and ResNet architecture and report two additional performance metrics. Here, we display the diagonal EF approximation due to its effectiveness (performance of Kronecker-factored variant in App. C.4). Additionally to accuracy and negative log likelihood, we show the expected calibration error (*ECE*, Guo et al. (2017)) and out-of-distribution detection performance in terms of area under the receiver-operator curve (*OOD-AUC*). With and without data augmentation, the models trained with our method achieve better test log-likelihood (*NLL*), expected calibration error (*ECE*), and out-of-distribution detection performance. Compared to the baseline, the performance can be up to $2\times$ better using our method. Notably, our method achieves a better NLL, ECE, and similar OOD-AUC *without* data augmentation than the baseline *with* data augmentation. However, the baseline can mitigate the difference in accuracy by data augmentation and match the performance on our method. Following Osawa et al. (2019), we upweight the likelihood by a factor of 5 to account for the data augmentation.

In Fig. 6, we show that the proposed online model selection can improve over a baseline with weight decay by greatly reducing overfitting and the generalization gap on CIFAR-10. Compared to a single training run that uses fixed hyperparameters, our online method only takes around twice the time for training when using the Kronecker-factored Laplace-GGN every $F = 10$ epochs. Cross-validation would only be able to explore two parameter settings in this time.

Finally, we conduct a larger-scale study for architecture

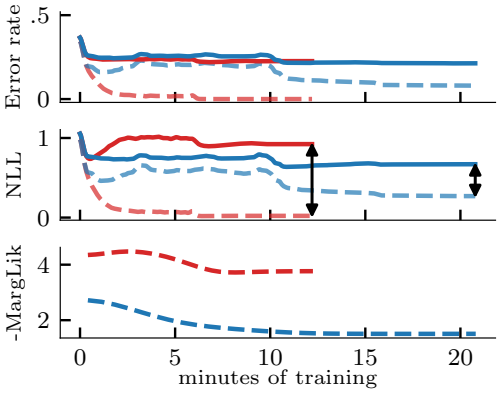

Figure 6: Train (- - -) vs. test (——) performance of our method (- - -) compared to a baseline with default weight decay (——) for a CNN on CIFAR-10. Online marginal likelihood optimization leads to a smaller generalization gap (◄►). While the test error rate matches, the baseline has a significantly worse negative log likelihood. The Laplace-GGN marginal likelihood estimate in the bottom indicates that as well. The Kronecker-factored Laplace-GGN approximation increases the training time only by a factor of 2. For cross-validation, this would only suffice to retrain for two different hyperparameters.

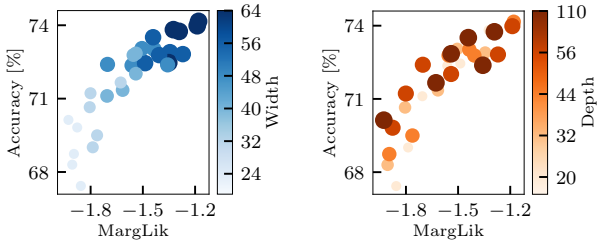

Figure 7: Wider ResNets give both higher marginal-likelihoods and accuracy (left) while there is no clear trend seen by varying depths (right). Results are on CIFAR-100. Each plot contains 30 models. Details in App. C.5

selection after training (step 2 in Sec. 3) by applying our algorithm to various architectures and comparing the resulting marginal likelihood estimates. We use the algorithm with the diagonal EF approximation due to its efficiency and optimize the prior precision hyperparameter for each layer.[1] After training, we use the final marginal likelihood estimates to rank the different architectures. On CIFAR-10 and CIFAR-100, we train CNNs and ResNets of varying width (between 2 and 64) and depth (between 1 and 110). On FashionMNIST, we compare CNNs and MLPs. For each data set, we compare around 40 models. In Fig. 2, each trained model with our method is represented by a marker whose size and color depends on the number of parameters. On CIFAR-10/100, ResNets achieve higher marginal likelihoods than CNNs despite, in some cases, exponentially more parameters. In terms of architecture, width tends to improve marginal likelihood more than depth (see Fig. 7 and App. C.5). This is in line with improved ResNet architectures that use increased width (Zagoruyko & Komodakis, 2016). On both data sets, the rank correlation between test accuracy and marginal likelihood is 97% in terms of Spearman's ρ. Spearman's ρ measures the correlation of the model rankings that test accuracy and marginal likelihood induce (Kendall, 1948). CNNs achieve consistently higher marginal likelihoods than MLPs on FMNIST. Especially at similar performance, CNNs achieve a higher marginal likelihood than MLPs, potentially due to the reduced model complexity. On FMNIST, the rank correlation is 69%. The correlation is likely lower because the marginal likelihood often selects significantly smaller instead of slightly better performing models (see top right in Fig. 2). For more detailed results and the architectures, see App. C.5.

[1] Similar results can be obtained with the Kronecker-factored variant but at a higher cost.

5. Conclusion

In this paper, we present the marginal likelihood as a viable tool for model selection in deep learning. Even after making approximations for scalability, the marginal likelihood faithfully reflects the model quality based on the training data alone. Estimation requires only a mild increase in computation, but allows us to compare models and even select hyperparameters during training. Ultimately, the marginal likelihood and similar methods make use of the neighborhood of a solution to improve their robustness. We believe this is important to design models for applications where little is known about deployment and future usage. Removing dependence on validation data can widen the impact of deep learning and open up new application areas.

Many interesting research directions remain open for exploration. It is important to explore ways to further improve Hessian approximations and computation, and consider other approximate inference methods. Online model selection for discrete hyperparameters is another interesting direction. Finally, extending this work beyond supervised learning, e.g., to bandits, Bayesian optimization, active learning, or continual learning, is an important avenue for future research. We hope that this work encourages others to pursue new ideas based on Bayesian model selection.

Acknowledgements

A. I. acknowledges funding by the Max Planck ETH Center for Learning Systems (CLS). M. B. acknowledges funding by the Max Planck Society, the EPSRC, and a Qualcomm European Scholarship in Technology. V. F. acknowledges funding by the Swiss Data Science Center.

References

Bishop, C. M. *Pattern recognition and machine learning*. Information Science and Statistics. Springer, 2006.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. Occam's razor. *Information processing letters*, 24 (6):377–380, 1987.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1613–1622, 2015.

Botev, A., Ritter, H., and Barber, D. Practical Gauss-Newton optimisation for deep learning. In *International Conference on Machine Learning*, International Convention Centre, Sydney, Australia, 2017. PMLR.

Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Buntine, W. L. and Weigend, A. S. Bayesian backpropagation. *Complex systems*, 5(6):603–643, 1991.

Damianou, A. and Lawrence, N. D. Deep Gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*. PMLR, 2013.

Dangel, F., Kunstner, F., and Hennig, P. Backpack: Packing more into backprop. In *Proceedings of 7th International Conference on Learning Representations*, 2019.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Dutordoir, V., van der Wilk, M., Artemev, A., and Hensman, J. Bayesian image classification with deep convolutional gaussian processes. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 2020.

Dziugaite, G. K. and Roy, D. M. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

Fong, E. and Holmes, C. On the marginal likelihood and cross-validation. *Biometrika*, 107(2):489–496, 2020.

Foong, A. Y., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. 'in-between'uncertainty in bayesian neural networks. *arXiv preprint arXiv:1906.11537*, 2019.

Foresee, F. D. and Hagan, M. T. Gauss-newton approximation to bayesian learning. In *International Conference on Neural Networks (ICNN'97)*, volume 3, pp. 1930–1935. IEEE, 1997.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2017.

Harville, D. A. Matrix algebra from a statistician's perspective, 1998.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869. PMLR, 2015.

Hochreiter, S. and Schmidhuber, J. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

Immer, A., Korzepa, M., and Bauer, M. Improving predictions of bayesian neural nets via local linearization. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pp. 703–711, 2021.

Jefferys, W. H. and Berger, J. O. Ockham's razor and bayesian analysis. *American Scientist*, 80(1):64–72, 1992. ISSN 00030996.

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., and Bengio, S. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2019.

Kendall, M. G. Rank correlation methods. 1948.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Khan, M., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. Fast and scalable bayesian deep learning by weight-perturbation in adam. In *International Conference on Machine Learning*, pp. 2611–2620, 2018.

Khan, M. E. E., Immer, A., Abedi, E., and Korzepa, M. Approximate inference turns deep networks into gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 3088–3098, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kristiadi, A., Hein, M., and Hennig, P. Being bayesian, even just a bit, fixes overconfidence in relu networks. In *International Conference on Machine Learning*, pp. 5436–5446. PMLR, 2020.

Kunstner, F., Hennig, P., and Balles, L. Limitations of the empirical fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*, pp. 4158–4169, 2019.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pp. 6402–6413, 2017.

Llorente, F., Martino, L., Delgado, D., and Lopez-Santiago, J. Marginal likelihood computation for model selection and hypothesis testing: an extensive review. *arXiv preprint arXiv:2005.08334*, 2020.

Lyle, C., Schut, L., Ru, R., Gal, Y., and van der Wilk, M. A bayesian perspective on training speed and model selection. *Advances in Neural Information Processing Systems*, 33, 2020.

MacKay, D. J. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

MacKay, D. J. Probable networks and plausible predictions—a review of practical bayesian methods for supervised neural networks. *Network: computation in neural systems*, 6(3):469–505, 1995.

MacKay, D. J. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pp. 13132–13143, 2019.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417, 2015.

Mascarenhas, W. F. The divergence of the bfgs and gauss newton methods. *Mathematical Programming*, 147(1): 253–276, 2014.

Neal, R. M. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

Neyman, J. and Pearson, E. S. Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933.

Osawa, K., Swaroop, S., Khan, M. E. E., Jain, A., Eschenhagen, R., Turner, R. E., and Yokota, R. Practical deep learning with bayesian principles. In *Advances in Neural Information Processing Systems*, pp. 4289–4301, 2019.

Rasmussen, C. E. and Ghahramani, Z. Occam's razor. In *Advances in neural information processing systems*, pp. 294–300, 2001.

Rasmussen, C. E. and Williams, C. K. *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006.

Rätsch, G., Onoda, T., and Müller, K.-R. Soft margins for adaboost. *Machine learning*, 42(3):287–320, 2001.

Ritter, H., Botev, A., and Barber, D. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

Robbins, H. *An empirical Bayes approach to statistics*. Office of Scientific Research, US Air Force, 1955.

Schneider, F., Balles, L., and Hennig, P. Deepobs: A deep learning optimizer benchmark suite. In *International Conference on Learning Representations*, 2018.

Snelson, E. L. *Flexible and efficient Gaussian process models for machine learning*. PhD thesis, UCL (University College London), 2007.

van der Wilk, M., Bauer, M., John, S., and Hensman, J. Learning invariances using the marginal likelihood. In *Advances in Neural Information Processing Systems*, pp. 9938–9948, 2018.

Wenzel, F., Roth, K., Veeling, B. S., Świątkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. How good is the bayes posterior in deep neural networks really? In *International Conference on Machine Learning*, 2020.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *BMVC*. BMVA Press, 2016.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pp. 5852–5861, 2018.

Zhang, G., Wang, C., Xu, B., and Grosse, R. B. Three mechanisms of weight decay regularization. In *ICLR (Poster)*. OpenReview.net, 2019a.

Zhang, H., Dauphin, Y. N., and Ma, T. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations*, 2019b.