Scalable Marginal Likelihood Estimation for Model Selection in Deep Learning (Appendix)

A. Derivations and additional details

A.1. Laplace approximation to the marginal likelihood

We briefly derive the Laplace approximation to the marginal likelihood and discuss the additional correction term that would arise in online estimation. We derive the local Laplace approximation at an arbitrary point $\boldsymbol{\theta}_*$ similar to the standard Laplace approximation by a local quadratic approximation. Recall the Hessian $\mathbf{H}_{\boldsymbol{\theta}} = -\nabla^2_{\theta\theta} \log p(\mathcal{D}, \theta|\mathcal{M})$ and define the gradient $\mathbf{g}_{\boldsymbol{\theta}} = -\nabla_\theta \log p(\mathcal{D}, \theta|\mathcal{M})$ of the MAP objective in Eq. 6. A second-order Taylor approximation around $\boldsymbol{\theta}_*$ to the log joint distribution is then given by

$$\log p(\mathcal{D}, \boldsymbol{\theta}|\mathcal{M}) \approx \log p(\mathcal{D}, \boldsymbol{\theta}_*|\mathcal{M}) - (\boldsymbol{\theta} - \boldsymbol{\theta}_*)^\intercal \mathbf{g}_{\boldsymbol{\theta}_*} - \tfrac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_*)^\intercal \mathbf{H}_{\boldsymbol{\theta}_*}(\boldsymbol{\theta} - \boldsymbol{\theta}_*). \tag{A.1}$$

In the standard Laplace approximation, the second term disappears due to $\mathbf{g}_{\boldsymbol{\theta}_*} = \mathbf{0}$. We compute the Laplace approximation to the marginal likelihood by solving the integral in Eq. 1 using the above approximation to the log joint:

$$\begin{aligned}
p(\mathcal{D}|\mathcal{M}) &= \int p(\mathcal{D}, \boldsymbol{\theta}|\mathcal{M}) \, \mathrm{d}\boldsymbol{\theta} \\
&\approx \int \exp\left[\log p(\mathcal{D}, \boldsymbol{\theta}_*|\mathcal{M}) - (\boldsymbol{\theta} - \boldsymbol{\theta}_*)^\intercal \mathbf{g}_{\boldsymbol{\theta}_*} - \tfrac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_*)^\intercal \mathbf{H}_{\boldsymbol{\theta}_*}(\boldsymbol{\theta} - \boldsymbol{\theta}_*)\right] \mathrm{d}\boldsymbol{\theta} \\
&= p(\mathcal{D}, \boldsymbol{\theta}_*|\mathcal{M}) \int \exp\left[-(\boldsymbol{\theta} - \boldsymbol{\theta}_*)^\intercal \mathbf{g}_{\boldsymbol{\theta}_*} - \tfrac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_*)^\intercal \mathbf{H}_{\boldsymbol{\theta}_*}(\boldsymbol{\theta} - \boldsymbol{\theta}_*)\right] \mathrm{d}\boldsymbol{\theta} \\
&= p(\mathcal{D}, \boldsymbol{\theta}_*|\mathcal{M}) \int \exp\left[-\tfrac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_* + \mathbf{H}^{-1}_{\boldsymbol{\theta}_*}\mathbf{g}_{\boldsymbol{\theta}_*})^\intercal \mathbf{H}_{\boldsymbol{\theta}_*}(\boldsymbol{\theta} - \boldsymbol{\theta}_* + \mathbf{H}^{-1}_{\boldsymbol{\theta}_*}\mathbf{g}_{\boldsymbol{\theta}_*}) + \tfrac{1}{2}\mathbf{g}^\intercal_{\boldsymbol{\theta}_*}\mathbf{H}^{-1}_{\boldsymbol{\theta}_*}\mathbf{g}_{\boldsymbol{\theta}_*}\right] \mathrm{d}\boldsymbol{\theta} \\
&= p(\mathcal{D}, \boldsymbol{\theta}_*|\mathcal{M}) \exp(\tfrac{1}{2}\mathbf{g}^\intercal_{\boldsymbol{\theta}_*}\mathbf{H}^{-1}_{\boldsymbol{\theta}_*}\mathbf{g}_{\boldsymbol{\theta}_*}) \int \exp\left[-\tfrac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_* + \mathbf{H}^{-1}_{\boldsymbol{\theta}_*}\mathbf{g}_{\boldsymbol{\theta}_*})^\intercal \mathbf{H}_{\boldsymbol{\theta}_*}(\boldsymbol{\theta} - \boldsymbol{\theta}_* + \mathbf{H}^{-1}_{\boldsymbol{\theta}_*}\mathbf{g}_{\boldsymbol{\theta}_*})\right] \mathrm{d}\boldsymbol{\theta} \\
&= p(\mathcal{D}, \boldsymbol{\theta}_*|\mathcal{M}) \exp(\tfrac{1}{2}\mathbf{g}^\intercal_{\boldsymbol{\theta}_*}\mathbf{H}^{-1}_{\boldsymbol{\theta}_*}\mathbf{g}_{\boldsymbol{\theta}_*})(2\pi)^{\frac{P}{2}} |\mathbf{H}_{\boldsymbol{\theta}_*}|^{-\frac{1}{2}}.
\end{aligned}$$

To obtain this result, we first complete the square inside the exponent:

$$\begin{aligned}
-(\boldsymbol{\theta} - \boldsymbol{\theta}_*)^\intercal \mathbf{g}_{\boldsymbol{\theta}_*} - \tfrac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_*)^\intercal \mathbf{H}_{\boldsymbol{\theta}_*}(\boldsymbol{\theta} - \boldsymbol{\theta}_*) &= -\left[(\boldsymbol{\theta} - \boldsymbol{\theta}_*)^\intercal \mathbf{g}_{\boldsymbol{\theta}_*} + \tfrac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_*)^\intercal \mathbf{H}_{\boldsymbol{\theta}_*}(\boldsymbol{\theta} - \boldsymbol{\theta}_*)\right] \\
&= -\left[\tfrac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_* + \mathbf{H}^{-1}_{\boldsymbol{\theta}_*}\mathbf{g}_{\boldsymbol{\theta}_*})^\intercal \mathbf{H}_{\boldsymbol{\theta}_*}(\boldsymbol{\theta} - \boldsymbol{\theta}_* + \mathbf{H}^{-1}_{\boldsymbol{\theta}_*}\mathbf{g}_{\boldsymbol{\theta}_*}) - \tfrac{1}{2}\mathbf{g}^\intercal_{\boldsymbol{\theta}_*}\mathbf{H}^{-1}_{\boldsymbol{\theta}_*}\mathbf{g}_{\boldsymbol{\theta}_*}\right].
\end{aligned} \tag{A.2}$$

The last step is to solve the Gaussian integral which gives the remaining factors $(2\pi)^{\frac{P}{2}} |\mathbf{H}_{\boldsymbol{\theta}_*}|^{-\frac{1}{2}}$. Taking the log, we obtain the log marginal likelihood approximation presented in Eq. 3 with an additional correction term $\frac{1}{2}\mathbf{g}^\intercal_{\boldsymbol{\theta}_*}\mathbf{H}^{-1}_{\boldsymbol{\theta}_*}\mathbf{g}_{\boldsymbol{\theta}_*}$. We empirically found that the correction term does not help during the online algorithm.For example see Fig. 6, where the marginal likelihood estimation remains stable during training without a correction term. However, the correction term could be computed as efficiently as the determinants we require for our algorithm. We believe the reason for its worse performance is due to an undesired change of the neural network parameter $\boldsymbol{\theta}_*$ to $\boldsymbol{\theta}_* + \mathbf{H}^{-1}_{\boldsymbol{\theta}_*}\mathbf{g}_{\boldsymbol{\theta}_*}$. The parameter change is due to the integration over $\boldsymbol{\theta}$ of Eq. A.2, which yields a local Laplace approximation *to the posterior* with mean $\boldsymbol{\mu} = \boldsymbol{\theta}_* + \mathbf{H}^{-1}_{\boldsymbol{\theta}_*}\mathbf{g}_{\boldsymbol{\theta}_*}$. This parameter update resembles a *full* step of Newton's method and does typically lead to divergence (Mascarenhas, 2014), except, for example, in linear least squares. We therefore ignore the correction term and treat the current parameter as $\boldsymbol{\theta}_*$. It is common to assume the gradient is zero, i.e., $\mathbf{g}_{\boldsymbol{\theta}_*} = \mathbf{0}$, when using the Laplace approximation in deep learning (Ritter et al., 2018; Kristiadi et al., 2020). We follow this simplification with our online approximation.

A.2. Efficient determinant computation

We can make the computation of the log determinant more efficient using Woodbury's identity and the same idea applies to the correction term in the local Laplace approximation. The local Laplace approximation relies on terms that involve expensive computations with the Hessian $\mathbf{H}_{\boldsymbol{\theta}}$ which we approximate either by the GGN or EF. That is, we have either $\mathbf{H} \approx \mathbf{J}_{\boldsymbol{\theta}}^\intercal \mathbf{L}_{\boldsymbol{\theta}} \mathbf{J}_{\boldsymbol{\theta}} + \mathbf{P}_{\boldsymbol{\theta}}$ (GGN) or $\mathbf{H} \approx \mathbf{G}_{\boldsymbol{\theta}}^\intercal \mathbf{G}_{\boldsymbol{\theta}} + \mathbf{P}_{\boldsymbol{\theta}}$ (EF). The results for the determinant are already presented in Eq. 8 and a proof can be found in the literature, e.g., see Theorem 18.1.1 by Harville (1998). For the correction term, we can similarly shift the computational complexity from the number of parameters to the number of data points. This can be achieved by standard application of Woodbury's matrix identity; see for example Theorem 18.2.8 by Harville (1998).

A.3. Kronecker-factored Laplace-GGN without damping

Kronecker-factored Laplace typically uses *damping* to combine a Gaussian prior with the Hessian approximation (Ritter et al., 2018). When using damping, the log likelihood Hessian and log prior Hessian are not added but also multiplied, which distorts the Laplace approximation. In particular because we optimize the prior parameters, such a distortion is likely to be problematic. We will first discuss damping and how we can compute the log determinant efficiently without damping, before presenting ablation results on image classification which suggest that damping should be avoided in this context.

Kronecker-factored Laplace approximates the GGN using two Kronecker factors, i.e., we have $[\mathbf{J}_{\boldsymbol{\theta}}^\intercal \mathbf{L}_{\boldsymbol{\theta}} \mathbf{J}_{\boldsymbol{\theta}}]_l \approx \mathbf{Q}_l \otimes \mathbf{W}_l$ where l denotes the index of a single layer. $\mathbf{Q}_l$ is quadratic in the number of neurons of the l-th layer while $\mathbf{W}_l$ is quadratic in the number of neurons of the previous layer (Botev et al., 2017). We assume an isotropic prior for the corresponding layer with $[\mathbf{P}_{\boldsymbol{\theta}}]_l = p_\theta^{(l)} \mathbf{I}_l$. Then, the Laplace approximation with damping yields

$$[\mathbf{H}_{\boldsymbol{\theta}}^{\text{GGN}}]_l \approx \mathbf{Q}_l \otimes \mathbf{W}_l + p_\theta^{(l)} \mathbf{I}_l \approx (\mathbf{Q}_l + \sqrt{p_\theta^{(l)}}\mathbf{I}) \otimes (\mathbf{W}_l + \sqrt{p_\theta^{(l)}}\mathbf{I}), \tag{A.3}$$

where $\mathbf{I}$ are of matching size. Clearly, the prior $p_\theta^{(l)}$ is now multiplied by the Kronecker factors and the update is not purely additive anymore. However, it is equally efficient to compute the determinant of the Kronecker approximation without damping. We use the eigendecomposition of $\mathbf{Q}_l \otimes \mathbf{W}_l$ which can be written as $\mathbf{M}_l(\operatorname{diag}(\mathbf{q}^{(l)}) \otimes \operatorname{diag}(\mathbf{w}^{(l)}))\mathbf{M}_l^\intercal$, where $\mathbf{M}_l$ is the eigenbase of the Kronecker product and $\mathbf{q}^{(l)}$ and $\mathbf{w}^{(l)}$ are vectors of eigenvalues of $\mathbf{Q}_l$ and $\mathbf{W}_l$ as defined before in Sec. 3. The $\operatorname{diag}(\cdot)$ operator turns a vector into a diagonal matrix with the corresponding vector as diagonal. The determinant for the GGN of one layer can be computed using properties of the Kronecker product and determinant:

$$\begin{aligned}
|[\mathbf{H}_{\boldsymbol{\theta}}^{\text{GGN}}]_l| &\approx |\mathbf{Q}_l \otimes \mathbf{W}_l + p_\theta^{(l)} \mathbf{I}_l| = |\mathbf{M}_l(\operatorname{diag}(\mathbf{q}^{(l)}) \otimes \operatorname{diag}(\mathbf{w}^{(l)}))\mathbf{M}_l^\intercal + p_\theta^{(l)} \mathbf{I}_l| \\
&= |\mathbf{M}_l(\operatorname{diag}(\mathbf{q}^{(l)}) \otimes \operatorname{diag}(\mathbf{w}^{(l)}) + p_\theta^{(l)} \mathbf{I}_l)\mathbf{M}_l^\intercal| = |\operatorname{diag}(\mathbf{q}^{(l)}) \otimes \operatorname{diag}(\mathbf{w}^{(l)}) + p_\theta^{(l)}| \\
&= \prod_{ij} \mathbf{q}_i^{(l)} \mathbf{w}_j^{(l)} + p_\theta^{(l)}.
\end{aligned}$$

To compute the final term, we only need the eigenvalues of both Kronecker factors and perform an outer product over the number of input and output neurons. The entire determinant is simply a product over the l layers due to the block-diagonal structure. The expression is given in Eq. 9.

We empirically compare the proposed version to the damped version and find that it is indeed necessary to use the exact version. The performance of the online algorithm using a damped Kronecker-factored Laplace-GGN yields significantly worse results with negative likelihoods up to two times worse. The accuracy suffers as well with a decrease of up to 4% points. Table A1 contains the results for the same experimental setup as described in Sec. 4 and Table 2.

Dataset	Model	KFAC accuracy	KFAC logLik	KFAC MargLik	KFAC *damped* accuracy	KFAC *damped* logLik	KFAC *damped* MargLik
MNIST	**MLP**	98.38±0.04	−0.053±0.002	−0.158±0.001	95.89±0.11	−0.141±0.004	−2.199±0.008
MNIST	**CNN**	99.46±0.01	−0.016±0.001	−0.064±0.000	98.78±0.09	−0.040±0.002	−1.022±0.017
FMNIST	**MLP**	89.83±0.14	−0.305±0.006	−0.468±0.002	84.12±0.23	−0.444±0.008	−2.022±0.026
FMNIST	**CNN**	92.06±0.10	−0.233±0.004	−0.401±0.001	89.54±0.10	−0.296±0.003	−2.424±0.013
CIFAR10	**CNN**	80.46±0.10	−0.644±0.010	−0.967±0.003	76.38±0.50	−0.690±0.014	−4.232±0.068

Table A1: Ablation of Kronecker-factored Laplace without and with *damping*: avoiding commonly used damping improves the performance across all experiments when using the online algorithm. Damping should be avoided in this context.

B. Computational details

We discuss the computational considerations necessary to apply the algorithm practically, i.e., to different hyperparameters, with different GGN and EF approximations, and number of hyperparameter updates K. In particular, we discuss the cost of line 6 for estimating the marginal likelihood and lines 7 to 10 for updating the hyperparameters and the marginal likelihood in Alg. 1 for individual Laplace-GGN and EF approximations and choices of differentiable hyperparameters. As detailed in Sec. 3.4, the other parameter settings need to be chosen to enable convergence of the marginal likelihood: the step size γ can be set by monitoring the convergence of the marginal likelihood and hyperparameters, and the update frequency F and number of burn-in epochs B should be set to ensure enough steps for convergence while keeping the computational overhead low. The settings F and B simply impact the number of marginal likelihood estimations.

Recall the neural network model introduced in Sec. 2: we have likelihood $p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M})$ and prior $p(\boldsymbol{\theta}|\mathcal{M})$, where the model $\mathcal{M}$ consists of hyperparameters that are differentiable in the marginal likelihood and discrete choices. Here, we entirely focus on the hyperparameters that are differentiable in the marginal likelihood. For both regression and classification, we work with an isotropic Gaussian prior per layer [2], i.e., $p(\boldsymbol{\theta}|\mathcal{M}) = \prod_l \mathcal{N}(\boldsymbol{\theta}_l|\mathbf{0}, \delta_l^{-1}\mathbf{I}_l)$ where l denotes the layer or group of parameters; we consider the weights and biases of each layer as individual parameter groups l. Hence, the Hessian of the log prior, $\mathbf{P}_{\boldsymbol{\theta}}$, is diagonal but with blocks of isotropic diagonals per layer l. For regression, we use a Gaussian likelihood and optimize the observation noise variance σ^2 using the marginal likelihood, i.e., we have the likelihood $p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M}) = \mathcal{N}(\mathbf{y}|\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}), \sigma^2)$. For classification, we only optimize the softmax temperature T in the illustrative example (Fig. 3) and have likelihood $p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M}) = \text{Categorical}(\mathbf{y}|\text{softmax}(\mathbf{f}(\mathbf{x}; \boldsymbol{\theta})/T))$ since it permits K cheap iterations without requiring re-computing expensive quantities.

We use either the GGN or EF approximation to the marginal likelihood in Eq. 3 at the current neural network parameters $\boldsymbol{\theta}_*$:

$$\log q(\mathcal{D}|\mathcal{M}) = \log p(\mathcal{D}|\boldsymbol{\theta}_*, \mathcal{M}) + \log p(\boldsymbol{\theta}_*|\mathcal{M}) + \tfrac{P}{2}\log 2\pi - \tfrac{1}{2}\log \left|\mathbf{H}^{\text{GGN/EF}}_{\boldsymbol{\theta}_*}\right|. \tag{B.1}$$

The first term, the log likelihood, can be computed based on the neural network functions $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_*)$ over the entire dataset in $\mathcal{O}(NP)$ but an unbiased estimator in $\mathcal{O}(MP)$ with minibatch M is also possible[3]. If we update hyperparameters of the log likelihood, for example, in regression, we further store the functions. The second term, the log prior, can be computed based on the current neural network parameters $\boldsymbol{\theta}_*$ in $\mathcal{O}(P)$ for our choice of prior. As outlined in Sec. 3, the major problem, and the only part in which the approximations differ, is to compute the GGN and EF log determinant. In the following, we discuss the cost of one determinant computation with the different approximations and the follow-up cost of K updates and hyperparameter steps. For classification, we deal with $\mathcal{M}^\partial = \{\delta_l\}$ and for regression we additionally have $\sigma^2 \in \mathcal{M}^\partial$.

B.1. Full Laplace-GGN

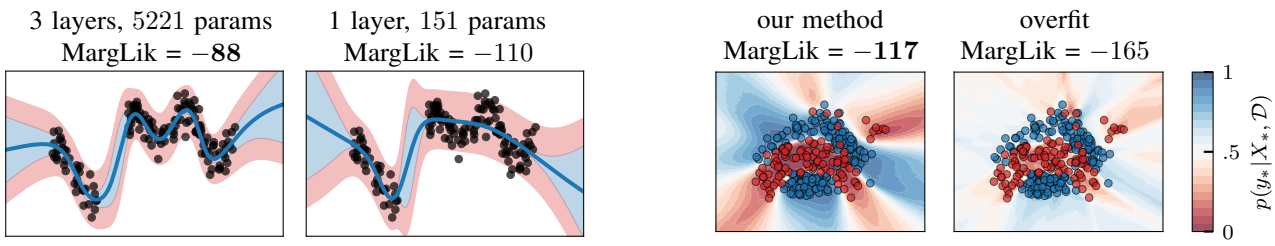
For the full Laplace-GGN, we need to pass through the training data to compute all Jacobians $\mathbf{J}_{\boldsymbol{\theta}}$ and log-likelihood Hessians $\mathbf{L}_{\boldsymbol{\theta}}$ with complexity $\mathcal{O}(NPC)$ and $\mathcal{O}(NC^2)$, respectively. We can then compute the log determinant in two ways depending on N, P, C. If $P < NC$, we use the standard method: we first compute the $P \times P$ matrix in Eq. 4 in $\mathcal{O}(P^2NC + PNC^2)$. Computing the log determinant of the resulting matrix is then $\mathcal{O}(P^3)$. Otherwise, i.e., if $NC < P$, we use the proposed formulation in Sec. 3, Eq. 8: in this case the computation is $\mathcal{O}(N^2C^2P)$ for construction and $\mathcal{O}(N^3C^3)$ to compute the determinant. Using multiple iterations are in general not cheaper unless we have an isotropic prior and $\mathbf{P}_{\boldsymbol{\theta}} \propto \mathbf{I}_P$, which is common in Bayesian deep learning. In this case, we can start with an initial eigendecomposition in $\mathcal{O}(P^3)$ or $\mathcal{O}(N^3C^3)$ but then update K times in $\mathcal{O}(P)$ and $\mathcal{O}(N)$, respectively.

B.2. Full Laplace-EF

In comparison to the full Laplace-GGN, the EF-based alternative scales independently of the number of outputs C and therefore typically cheaper and we can trade off between P and N directly. We first need to compute (and store) the individual gradients in $\mathcal{O}(NP)$. If $P < N$, we use the standard version by computing Eq. 5 in $\mathcal{O}(P^2N)$ and then the determinant in $\mathcal{O}(P^3)$. If $N < P$, we use the matrix determinant equivalence in Eq. 8 to reduce computation to $\mathcal{O}(N^2P)$ and determinant computation to $\mathcal{O}(N^3)$. In line with the full Laplace-GGN, multiple iterations are in general not cheaper unless $\mathbf{P}_{\boldsymbol{\theta}} \propto \mathbf{I}_P$ in which case we can use an initial eigendecomposition of the same complexity and iterate cheaply in $\mathcal{O}(P)$ or $\mathcal{O}(N)$.

[2] Except for Fig. 4 where we compare against a grid-search which is expensive already for two dimensions and high resolution.

[3] We assume a standard architecture that allows a forward pass in $\mathcal{O}(P)$



(a) Bayesian predictive for regression example.

(b) Bayesian predictive for classification example.

Figure C.1: Optional Bayesian posterior predictives (Eq. 11) as alternative to the MAP predictives (Eq. 12) displayed in Fig. 1 and Fig. 3. (a) Posterior predictive for regression example with noise variance ( ) and neural network model uncertainty ( ). Marginal likelihood training results in Bayesian predictives with a reasonable split into both uncertainties which is hard to obtain by other means. (b) The posterior predictive for classification example yields a predictive that is certain in the data region and less certain away from it when using marginal likelihood optimization.

B.3. Kronecker-factored Laplace-GGN or EF

The Kronecker-factored Laplace approximations are not only cheaper to compute and store but are also significantly more efficient in updating per-layer hyperparameters due to the per-layer block-diagonal approximation. During the pass through the training data, the Kronecker factors are computed and stored (Botev et al., 2017) which is $\mathcal{O}(NP+N(D^2+C^2+\sum_l H_l^2))$ where H_l is the size of the l-th hidden layer. That is because each Kronecker factor is quadratic in the number of neurons of the input or output of some layer. We compute the eigendecomposition of all Kronecker factors in $\mathcal{O}(D^3C^3 + \sum_l H_l^3)$ but only keep the eigenvalues necessary to compute Eq. 9. Updating the per-layer prior precision and observation noise variance are only $\mathcal{O}(P)$ due to Eq. 9 and many hyperparameter updates help amortize the up-front cost. We compute the Kronecker factors using the backpack package by Dangel et al. (2019).

B.4. Diagonal Laplace-GGN

The diagonal Laplace-GGN requires backpropagation of $C \times C$ matrices $\mathbf{\Lambda}(\mathbf{y}; \mathbf{f})$ (Dangel et al., 2019) and is therefore not as fast as the diagonal EF. The determinant computation, however, is only $\mathcal{O}(P)$. Once the diagonal GGN is obtained, hyperparameter updates are as cheap as in the Kronecker-factored case in $\mathcal{O}(P)$ including updates to the determinant. Empirically, the diagonal Laplace-GGN is not significantly faster than the Kronecker-factored Laplace but it makes a different approximation and therefore might under some circumstances work better (see App. C.4).

B.5. Diagonal Laplace-EF

The diagonal Laplace-EF is cheaper and simpler to compute than the diagonal Laplace-GGN while maintaining the same hyperparameter update complexity of $\mathcal{O}(P)$. To compute the EF while passing through the training data, we only need to compute individual gradients, square them element-wise, and sum them up. Hence, this is as cheap as a pass through the training data for neural network parameters (roughly $\mathcal{O}(NP)$). One hyperparameter update is then $\mathcal{O}(P)$.

C. Experimental details and additional results

C.1. Illustrative examples

We complement the results of Sec. 4.1 with Bayesian predictives obtained after running our online algorithm.

Bayesian regression predictive. For a Gaussian likelihood, the Bayesian predictive (cf. Eq. 11) has a closed-form and we do not need samples (Foong et al., 2019; Khan et al., 2019; Immer et al., 2021). We additionally show the Bayesian predictive and its decomposition in epistemic and aleatoric uncertainty for the illustrative regression example in Fig. 3. The aleatoric, irreducible, uncertainty is the noise in the data. The epistemic uncertainty is model-dependent and quantifies how certain a model is about its prediction. In the regression example, the aleatoric uncertainty is the observation noise σ^2 that we learn online and the epistemic uncertainty is the model uncertainty $\mathrm{Var}[\mathbf{f}]$ which we optimize implicitly via the prior. The variance of the posterior predictive is then $\mathrm{Var}[\mathbf{f}] + \sigma^2$ and the standard deviation is not additive. In Fig. C.1a, we split

Dataset	cross-validation MAP	cross-validation VI	cross-validation Laplace	MargLik optimization GGN-Bayes	MargLik optimization GGN-MAP	MargLik optimization EF-Bayes	MargLik optimization EF-MAP	MargLik optimization KFAC-Bayes	MargLik optimization KFAC-MAP
boston	2.71±0.10	**2.58±0.03**	**2.57±0.05**	2.68±0.12	2.69±0.13	**2.60±0.09**	2.62±0.12	2.70±0.08	2.70±0.09
concrete	3.17±0.04	3.17±0.01	**3.05±0.04**	**3.06±0.05**	**3.06±0.05**	**3.07±0.04**	**3.07±0.04**	3.13±0.06	3.13±0.06
energy	1.02±0.05	1.42±0.01	0.82±0.03	**0.54±0.11**	**0.55±0.11**	0.78±0.13	0.73±0.15	**0.54±0.02**	**0.53±0.02**
kin8nm	−1.09±0.01	−0.87±0.00	**−1.23±0.01**	−1.14±0.01	−1.14±0.01	−1.13±0.01	−1.14±0.01	−1.13±0.01	−1.13±0.01
naval	−5.75±0.05	−3.82±0.02	−6.40±0.06	−6.61±0.03	**−6.93±0.03**	−6.21±0.05	**−6.92±0.04**	−6.32±0.05	−6.88±0.05
power	2.82±0.01	2.86±0.01	2.83±0.01	**2.78±0.02**	**2.78±0.02**	**2.78±0.01**	**2.78±0.01**	**2.78±0.02**	**2.78±0.02**
wine	0.98±0.02	0.96±0.01	0.97±0.02	**0.94±0.02**	**0.93±0.02**	**0.94±0.01**	**0.93±0.02**	**0.94±0.02**	**0.94±0.02**
yacht	2.30±0.02	1.67±0.01	**1.01±0.05**	3.51±0.62	5.89±1.25	1.29±0.17	2.43±0.61	1.48±0.07	1.48±0.07

Table C2: Additional performance of the Bayesian posterior predictive on the UCI regression benchmark complementing Table 1. The posterior predictive does not significantly improve the predictive except on the yacht data set which is very small and might require better uncertainty quantification.

Dataset	GGN-Bayes	GGN-MAP	diag-GGN-Bayes	diag-GGN-MAP	diag-EF-Bayes	diag-EF-MAP
boston	**2.68±0.12**	**2.69±0.13**	2.84±0.05	2.84±0.05	2.90±0.03	2.89±0.03
concrete	**3.06±0.05**	**3.06±0.05**	3.13±0.04	3.13±0.04	**3.11±0.02**	**3.11±0.02**
energy	**0.54±0.11**	**0.55±0.11**	0.70±0.03	0.69±0.03	0.88±0.05	0.85±0.05
kin8nm	**−1.14±0.01**	**−1.14±0.01**	**−1.14±0.01**	**−1.13±0.01**	**−1.13±0.01**	**−1.13±0.01**
naval	−6.61±0.03	**−6.93±0.03**	−6.17±0.05	**−6.91±0.08**	−5.83±0.06	**−6.95±0.04**
power	**2.78±0.02**	**2.78±0.02**	**2.79±0.02**	**2.79±0.02**	**2.79±0.01**	**2.79±0.01**
wine	**0.94±0.02**	**0.93±0.02**	0.96±0.01	**0.95±0.02**	0.98±0.01	0.96±0.02
yacht	3.51±0.62	5.89±1.25	**2.91±0.04**	**2.90±0.04**	3.16±0.02	3.15±0.02

Table C3: Performance on UCI regression tasks of the diagonal EF and GGN online marginal likelihood optimization method compared to the full GGN which performs best. The diagonal approximation slightly decreases the performance but is not significantly worse. In comparison to a cross-validated MAP, the cheap diagonal approximations still perform quite well (cf. first column in Table C2).

up the total predictive variance approximately so that the observation noise σ is depicted exactly and the model uncertainty is just the remaining standard deviation.

Bayesian classification predictive. Fig. C.1b depicts the Bayesian predictives corresponding to the schematic figure Fig. 3. We use $S = 1000$ samples to estimate the posterior predictive in Eq. 11. The samples are on the output of the network and are therefore cheap (Immer et al., 2021). The Bayesian predictive of the model with a better marginal likelihood has increasing uncertainty away from the data as often desired (Foong et al., 2019). The parameters found by our online algorithm give rise to a Bayesian predictive without further tuning of parameters after training which is required when using the Laplace approximation (Ritter et al., 2018; Kristiadi et al., 2020). Alternatively, one can run a grid-search to find suitable parameters (Khan et al., 2019; Immer et al., 2021). Our algorithm is more principled than changing the prior after training and significantly more efficient than a grid-search.

Regression grid search. In Fig. 4, we compared a grid-search over a single prior parameter with the online optimization algorithm and show it converges to the optimum marginal likelihood with different initializations. Here, we additionally show in Fig. C.2 that the best marginal likelihood also corresponds to the best negative log likelihood (nll) on the held-out test set. The online optimization algorithm reliably finds the optimum without access to the test data.

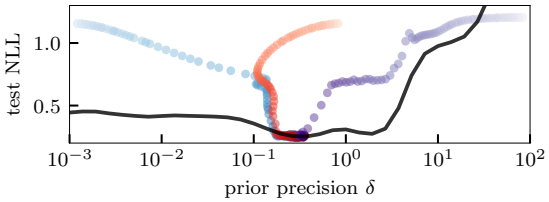


Figure C.2: Test negative log likelihood versus prior precision δ corresponding to Fig. 4. The proposed method reliably converges to the optimal predictive performance.

C.2. UCI regression

We complement the results presented in Sec. 4.2 and in Table 1 with the performance of the optional Bayesian posterior predictive and of the diagonal determinant approximations. We always report the performance of the posterior predictive using the posterior of the same structure as the determinant approximation during marginal likelihood optimization, e.g., if we use the full EF during optimization, we use the Laplace-EF posterior and make predictions with it using Eq. 11. In Table C2, we report the performance of the additional posterior predictive for the full GGN, EF, and KFAC determinant computations and their corresponding posterior predictive. The MAP predictive overall seems to perform better except on the small yacht data set. Table C3 further shows the performance when using the diagonal GGN or EF approximations.

Dataset	cross-validation MAP	cross-validation VI	cross-validation Laplace	MargLik optimization GGN-Bayes	MargLik optimization GGN-MAP	MargLik optimization EF-Bayes	MargLik optimization EF-MAP	MargLik optimization KFAC-Bayes	MargLik optimization KFAC-MAP
australian	**0.32±0.01**	**0.32±0.01**	**0.32±0.01**	**0.31±0.01**	**0.31±0.01**	0.34±0.01	**0.31±0.01**	**0.32±0.01**	**0.31±0.01**
cancer	0.13±0.03	0.17±0.05	0.10±0.01	0.10±0.01	**0.09±0.01**	0.25±0.01	0.10±0.02	0.10±0.01	**0.09±0.01**
ionosphere	0.28±0.02	**0.23±0.02**	0.28±0.01	0.29±0.02	0.26±0.03	0.42±0.01	0.28±0.04	0.31±0.02	0.29±0.02
glass	**0.89±0.03**	1.07±0.07	**0.88±0.01**	0.92±0.02	0.91±0.05	1.37±0.02	1.03±0.10	0.98±0.02	0.98±0.04
vehicle	0.39±0.01	**0.37±0.01**	0.46±0.00	0.47±0.01	0.41±0.01	0.75±0.01	**0.38±0.01**	0.49±0.00	0.45±0.01
waveform	0.36±0.00	**0.35±0.00**	0.37±0.00	**0.35±0.01**	**0.34±0.01**	0.43±0.01	0.36±0.02	**0.35±0.01**	**0.34±0.01**
digits	**0.08±0.00**	0.14±0.01	0.26±0.00	0.31±0.01	0.13±0.01	1.81±0.01	**0.07±0.01**	0.30±0.01	0.16±0.01
satellite	**0.23±0.00**	0.28±0.00	0.25±0.00	0.25±0.00	**0.22±0.01**	0.68±0.01	0.24±0.01	0.25±0.00	0.24±0.00

(a) Negative test log likelihood (lower better)

Dataset	cross-validation MAP	cross-validation VI	cross-validation Laplace	MargLik optimization GGN-Bayes	MargLik optimization GGN-MAP	MargLik optimization EF-Bayes	MargLik optimization EF-MAP	MargLik optimization KFAC-Bayes	MargLik optimization KFAC-MAP
cancer	**96.8±0.4**	96.4±0.4	**97.0±0.3**	**96.8±0.4**	**96.9±0.4**	**96.7±0.5**	**96.7±0.5**	**96.7±0.5**	**96.7±0.5**
ionosphere	90.6±0.7	**92.6±0.7**	90.6±0.6	90.0±1.3	90.0±1.3	90.9±1.2	91.5±1.0	89.1±1.3	88.9±1.4
glass	**67.2±1.4**	63.4±1.4	65.3±1.4	64.4±2.7	64.1±2.8	**65.9±2.7**	**67.8±2.2**	63.7±2.0	64.7±1.9
vehicle	**83.2±0.3**	81.4±0.4	80.9±0.3	81.7±0.9	82.0±0.9	82.1±0.6	**83.2±0.5**	80.3±0.8	80.3±0.8
waveform	**85.7±0.3**	**85.6±0.3**	**85.7±0.3**	**85.9±0.6**	**85.9±0.5**	84.7±0.8	84.3±0.8	**86.0±0.5**	**85.9±0.6**
digits	**98.1±0.1**	97.3±0.1	97.1±0.1	97.3±0.2	97.4±0.2	86.9±0.8	**98.1±0.2**	97.0±0.4	97.1±0.3
satellite	**91.6±0.1**	89.9±0.1	91.3±0.1	**91.6±0.2**	**91.6±0.3**	91.1±0.4	**91.5±0.3**	91.0±0.2	91.0±0.2

(b) Test accuracy (higher better)

Dataset	cross-validation MAP	cross-validation VI	cross-validation Laplace	MargLik optimization GGN-Bayes	MargLik optimization GGN-MAP	MargLik optimization EF-Bayes	MargLik optimization EF-MAP	MargLik optimization KFAC-Bayes	MargLik optimization KFAC-MAP
australian	**0.06±0.01**	**0.06±0.00**	0.07±0.01	0.07±0.01	**0.05±0.01**	0.09±0.01	**0.06±0.00**	**0.06±0.01**	**0.06±0.01**
cancer	**0.03±0.00**	**0.03±0.00**	**0.03±0.00**	**0.03±0.00**	**0.03±0.00**	0.18±0.01	**0.03±0.00**	**0.03±0.00**	**0.03±0.00**
ionosphere	0.08±0.00	0.07±0.00	0.11±0.00	0.10±0.01	0.08±0.01	0.22±0.01	**0.06±0.01**	0.10±0.01	0.08±0.01
glass	**0.17±0.01**	**0.17±0.01**	0.19±0.01	**0.18±0.02**	**0.17±0.01**	0.36±0.02	**0.18±0.02**	0.19±0.02	**0.18±0.01**
vehicle	**0.06±0.00**	0.07±0.00	0.10±0.00	0.12±0.01	0.07±0.01	0.30±0.01	**0.06±0.01**	0.12±0.01	0.08±0.01
waveform	0.05±0.00	**0.04±0.00**	0.06±0.00	0.07±0.00	0.05±0.01	0.12±0.01	0.05±0.00	0.07±0.00	0.05±0.00
digits	**0.01±0.00**	0.03±0.00	0.17±0.00	0.20±0.00	0.06±0.00	0.70±0.01	**0.01±0.00**	0.19±0.00	0.08±0.00
satellite	**0.02±0.00**	**0.02±0.00**	0.04±0.00	0.05±0.00	**0.02±0.00**	0.36±0.00	**0.02±0.00**	0.05±0.00	0.03±0.00

(c) Test expected calibration error (lower better)

Dataset	GGN-Bayes	GGN-MAP	diag-GGN-Bayes	diag-GGN-MAP	diag-EF-Bayes	diag-EF-MAP
australian	**0.31±0.01**	**0.31±0.01**	0.35±0.01	0.33±0.01	0.34±0.01	**0.32±0.01**
cancer	0.10±0.01	**0.09±0.01**	0.12±0.01	**0.09±0.01**	0.16±0.01	**0.09±0.01**
ionosphere	**0.29±0.02**	**0.26±0.03**	0.38±0.01	0.35±0.01	0.37±0.01	0.31±0.02
glass	**0.92±0.02**	**0.91±0.05**	1.09±0.01	1.06±0.02	1.05±0.02	0.99±0.03
vehicle	0.47±0.01	**0.41±0.01**	0.75±0.01	0.68±0.01	0.71±0.01	0.59±0.01
waveform	**0.35±0.01**	**0.34±0.01**	0.41±0.01	0.38±0.01	0.40±0.01	0.37±0.01
digits	0.31±0.01	0.13±0.01	0.43±0.00	0.23±0.01	0.97±0.02	**0.09±0.01**
satellite	0.25±0.00	**0.22±0.01**	0.34±0.00	0.30±0.00	0.35±0.00	0.29±0.00

(d) Negative test log likelihood (lower better) of diag-GGN and diag-EF compared to full GGN.

Table C4: Performance of the proposed online marginal likelihood optimization method compared to cross-validation on UCI classification benchmark using a `ReLU` network with 50 units and one hidden layer. The marginal likelihood optimization leads to comparable performance as cross-validation and the MAP typically performs better than Laplace or VI. The full EF approximation performs best, followed by the GGN and KFAC approximations. Table C4d shows that the diagonal approximations perform worse, especially whem combined with a Bayesian predictive, but are still on a similar level as VI with cross-validation. Results within one standard error are in bold.

C.3. UCI classification

We present additional results on small-scale UCI classification data sets in Table C4. The setup is identical to that in the regression case but we use a different train/validation/test split of $70\%/15\%/15\%$ here. We use the same architecture as in the regression case, a single hidden layer with 50 units and `ReLU` activation and train for $10,000$ iterations until convergence with all methods. For the grid-search we try 10 different scalar prior precision values δ and select the best parameter on the validation set before re-training. We compare the resulting cross-validated performance to our online algorithm with full GGN, EF, and KFAC approximation to the determinant in Table C4. All methods perform quite well in this benchmark and there are no obvious differences in the negative log likelihood (nll) in Table C4a. However, the cross-validated MAP and full EF-MAP perform better than most other methods in terms of accuracy and expected calibration error.

C.4. Image Classification: online algorithm compared to cross-validation

We provide additional details and the following additional results on the image classification experiments presented in Sec. 4.3: the standard errors for Table 2, the performance of the diagonal GGN, and a comparison of different ResNet depths on CIFAR-10.

Architectures. We compare a fully-connected, convolutional, and residual neural network in our experiments. The fully-connected network has four hidden layers with decreasing hidden layer sizes $1024, 512, 256, 128$. For MNIST and FMNIST, this architecture has $P = 1,494,154$ parameters. As a standard convolutional neural network, we use a network of three convolutional layers followed by three fully-connected layers. This network is used in standard benchmarks, for example the suite by Schneider et al. (2018). On MNIST and FMNIST, this architecture has $P = 892,010$ parameters and on CIFAR-10 it has $P = 895,210$ parameters due to the additional color channels in the input. On CIFAR-10, we additionally use a ResNet-20 architecture with $P = 268,393$ parameters using the Fixup parameterization (Zhang et al., 2019b)[4]. We find that increasing the depth of the ResNet up to a ResNet-110 did not improve the performance and marginal likelihood significantly as we show below in Table C8. There, we additionally use ResNets with depths $32, 44, 56, 110$ which have $P = 461,959$, $P = 655,525$, $P = 849,091$, and $P = 1,720,138$ parameters, respectively.

Diagonal GGN and standard errors. Table C5 contains the additional standard errors that were left out in the main text in Table 2 due to space constraints. In Table C6, we compare the performance of our method using the diagonal EF and the diagonal GGN. The latter has been left out in the image classification benchmark in the main text. The table shows that the EF works consistently better. Note that the EF is also significantly cheaper to compute and easier to implement than the other methods.

		cross-validation		MargLik optimization					
				KFAC			diagonal EF		
Dataset	Model	accuracy	logLik	accuracy	logLik	MargLik	accuracy	logLik	MargLik
MNIST	MLP	98.22±0.13	−0.061±0.004	98.38±0.04	−0.053±0.002	−0.158±0.001	97.05±0.09	−0.095±0.002	−0.553±0.021
	CNN	99.40±0.03	**−0.017±0.001**	**9.46±0.01**	**−0.016±0.001**	**−0.064±0.000**	**9.45±0.03**	−0.019±0.001	−0.134±0.001
FMNIST	MLP	88.09±0.10	−0.347±0.005	89.83±0.14	−0.305±0.006	−0.468±0.002	85.72±0.09	−0.400±0.003	−0.756±0.005
	CNN	91.39±0.11	−0.258±0.004	**92.06±0.10**	**−0.233±0.004**	**−0.401±0.001**	91.69±0.15	**−0.233±0.003**	−0.570±0.003
CIFAR10	CNN	77.41±0.06	−0.680±0.004	80.46±0.10	−0.644±0.010	−0.967±0.003	80.17±0.29	−0.600±0.010	−1.359±0.010
	ResNet	83.73±0.28	−1.060±0.022	**86.11±0.39**	−0.595±0.017	**−0.717±0.003**	**85.82±0.13**	**−0.464±0.007**	−0.876±0.012

Table C5: Standard errors for Table 2 in the main text. Best performances within one standard error per dataset in bold.

			diagonal EF			diagonal GGN	
Dataset	Model	accuracy	logLik	MargLik	accuracy	logLik	MargLik
MNIST	MLP	97.05±0.09	−0.095±0.002	−0.553±0.021	96.86±0.04	−0.102±0.002	−0.617±0.014
	CNN	99.45±0.03	−0.019±0.001	−0.134±0.001	99.35±0.02	−0.021±0.001	−0.193±0.001
FMNIST	MLP	85.72±0.09	−0.400±0.003	−0.756±0.005	85.36±0.21	−0.410±0.006	−0.844±0.016
	CNN	91.69±0.15	−0.233±0.003	−0.570±0.003	91.69±0.09	−0.237±0.003	−0.641±0.002
CIFAR10	CNN	80.17±0.29	−0.600±0.010	−1.359±0.010	79.98±0.46	−0.587±0.013	−1.568±0.010

Table C6: Comparison of diagonal GGN and EF on the image classification benchmark. The diagonal EF as presented in Table 2 performs slightly better on all datasets and is cheaper to compute.

KFAC results for CIFAR-10 with data augmentation. We presented the performance of the marginal likelihood optimization using diagonal EF on CIFAR-10 with and without data augmentation (DA) in Fig. 5. In Table C7, we provide the results in terms of average performance over five random seeds and the performance of the KFAC GGN variant. On this benchmark, we see that the cheaper diagonal EF performs typically slightly better while being more efficient than the KFAC variant.

Deeper ResNets. In the main text, we only presented results for a ResNet-20. That is because we found that a depth over 20 for CIFAR-10 did not improve the marginal likelihood or performance noticeably. Here, we briefly present the corresponding results with and without data augmentation. Interestingly, the optimized marginal likelihood is largely unaffected by the almost 10-fold increase of parameters. In Table C8, we show the performances, marginal likelihoods, and

[4]Their implementation is publicly available with the corresponding standard settings on CIFAR-10: https://github.com/hongyi-zhang/Fixup

Dataset	Method	accuracy	logLik	ECE	OOD-AUC
CIFAR-10	**baseline**	83.73±0.28	−1.06±0.02	0.127±0.002	0.814±0.007
CIFAR-10	**MargLik diag EF**	85.82±0.13	−0.46±0.01	0.048±0.002	0.901±0.004
CIFAR-10	**MargLik KFAC GGN**	86.11±0.39	−0.60±0.02	0.085±0.003	0.882±0.008
CIFAR-10 + DA	**baseline**	91.38±0.15	−0.58±0.01	0.069±0.001	0.897±0.003
CIFAR-10 + DA	**MargLik diag EF**	91.18±0.20	−0.38±0.02	0.054±0.003	0.900±0.003
CIFAR-10 + DA	**MargLik KFAC GGN**	91.26±0.12	−0.54±0.02	0.068±0.001	0.896±0.006

Table C7: Results displayed in Fig. 5 with additional results for the marginal likelihood training based on KFAC GGN.

parameters with standard errors over five initializations. Without data augmentation, the smaller models are preferred. With data augmentation, the medium-sized models fare best in terms of marginal likelihood.

		no DA			with DA		
Model	**P**	**accuracy**	**logLik**	**MargLik**	**accuracy**	**logLik**	**MargLik**
ResNet-20	268,393	85.82±0.13	−0.46±0.01	−0.876±0.012	91.18±0.20	−0.375±0.020	−0.633±0.002
ResNet-32	461,959	85.66±0.29	−0.46±0.01	−0.898±0.025	91.63±0.22	−0.355±0.017	−0.623±0.008
ResNet-44	655,525	85.63±0.24	−0.45±0.01	−0.904±0.014	91.78±0.11	−0.336±0.007	−0.622±0.009
ResNet-56	849,091	85.82±0.17	−0.44±0.01	−0.932±0.010	91.62±0.15	−0.344±0.012	−0.625±0.013
ResNet-110	1,720,138	86.11±0.24	−0.44±0.01	−0.937±0.009	91.85±0.13	−0.323±0.009	−0.638±0.009

Table C8: ResNets of various depths trained on CIFAR-10 and their sizes, performances, and marginal likelihoods with and without data augmentation (DA). Without data augmentation, the smallest model achieves the highest marginal likelihood and the performances of all models are very close to each other. With data augmentation, the marginal likelihoods overlap in terms of their standard errors but a ResNet-44 seems best. Notably, the increase of the number of parameters P does not significantly impact the marginal likelihood.

C.5. Image Classification: architecture selection using the marginal likelihood

We presented results on architecture selection using MargLik in Sec. 4.3, Fig. 2, and Fig. 7. Fig. 7 is the upper right of Fig. 2 and only shows the ResNets colored by width and depth. Here, we describe details on the experimental setup and architectures, and list more detailed results.

Model training. All models where trained using Alg. 1 for 250 epochs, frequency $F = 5$, $K = 100$ hyperparameter updates with learning rate $\gamma = 1$, and no burnin $B = 0$. We use the ResNet learning rate decay as explained in Sec. 4 for all models and train with SGD with initial learning rate of 0.01 and a batch-size of 128.

Architectures. On FMNIST we compare MLPs and CNNs, and on CIFAR we compare CNNs and ResNets. For the MLPs, we use 3 different widths $50, 200, 800$ and depth from 1 to 5. The CNNs consist of up to 5 blocks of 3×3 convolutions, followed by a ReLU operation, and MaxPooling, except in the first layer. Instead of BatchNorm, we use the fixup parameterization (Zhang et al., 2019b) after every convolutional layer. For the widths, we consider widths, i.e., number of channels, from 2 to 32 in the first convolution. Each following convolution uses $2\times$ more channels such that after three convolutions the number of channels is $2^3 = 8$ times higher. The last layer is a fully-connected layer to the class logits. We use ResNets of depths from 8 to 32 on CIFAR-10 and from 20 to 101 on CIFAR-100. On CIFAR-10, more depth does not further increase or change the marginal likelihood, see Table C8. For the width, we vary the number of channels in the first convolutional layer. For CIFAR-10, we consider widths from 16 to 48 and on CIFAR-10 from 32 to 64. All the architectures on the corresponding data sets are listed with the marginal likelihood, accuracy, number of parameters, width, and depth in Table C11 (FMNIST), Table C9 (CIFAR-10), and Table C10 (CIFAR-100).

Additional figures. The markers in the scatter plots of architectures are colored and scaled by the number of parameters in Fig. 2. Here, we additionally show a coloring and scaling by width and depth in Fig. C.3 and Fig. C.4, respectively. Fig. C.3 shows that width tends to increase marginal likelihood and test accuracy on CIFAR. This is in line with improved ResNet architectures that proposed increasing the width (Zagoruyko & Komodakis, 2016). On FMNIST, the comparison between MLPs and CNNs is not meaningful in terms of width since width on CNNs determines the number of channels instead of hidden units. However, we can see that less wide MLPs achieve a better marginal likelihood and test accuracy. Fig. C.4 shows that increased depth does increase the marginal likelihood but saturates much earlier than width. For example, ResNets, which are deeper, are generally preferred over CNNs on CIFAR but the best models are not the deepest ResNets, see also Table C9 and Table C10. On FMNIST, we see in Fig. C.4 how the depth tends to decrease marginal likelihood and test accuracy of MLPs. Contrary to that, CNNs profit significantly from increasing the depth.

C.6. Robustness to algorithm hyperparameters

Alg. 1 has four different hyperparameters that need to be set. K determines how many gradient steps we take on the marginal likelihood objective. F determines how frequently we compute the marginal likelihood and optimize it with respect to hyperparameters. B determines for how many epochs we do not optimize the marginal likelihood, similar to a burn-in period in MCMC algorithms. On CIFAR-10, we conduct an ablation experiment running over a grid of these hyperparameters. Below, we make recommendations on how to apply our algorithm based on the outcomes of this experiment.

Step size γ. The step size γ only needs to be set to ensure the marginal likelihood objective does not diverge. γ does not affect the runtime but only convergence behavior. We find that using ADAM with default learning rate of $\gamma = 0.001$ works well but we can increase the learning rate without divergence to $\gamma = 1$ which ultimately decreases the number of steps K we need to take every F-th epoch to observe convergence. To set γ, a practitioner only needs to monitor the marginal likelihood or the parameters and avoid divergence or oscillation.

Frequency and number of steps. Number of steps K, frequency F, and burn-in steps B can be used to reduce the runtime by avoiding to compute the Laplace approximation too often. B is only a mechanism to reduce the runtime in the early training regime where the parameters of the network $\boldsymbol{\theta}$ are very noisy. If affordable, $B = 0$ is optimal for performance but we find that $B = 50$ works equivalently well. In many cases, convergence of the neural network training can even be improved by adjusting hyperparameters early. K and F determine how many hyperparameter steps we take and how many Laplace approximations need to be computed, respectively. Computing the Laplace approximation is expensive while hyperparameter steps are *amortized*, i.e., once we computed the Laplace approximation it is cheap to do many gradient steps K (cf. App. B). It is therefore ideal for performance to have $F = 1$ and compute the Laplace approximation after every epoch and have a large number of steps K. In practice, however, we find that F between 5 and 10 is sufficient and does not decrease performance. Further, $K \geq 100$ also appears to suffice for convergence with no gain using $K = 1000$.

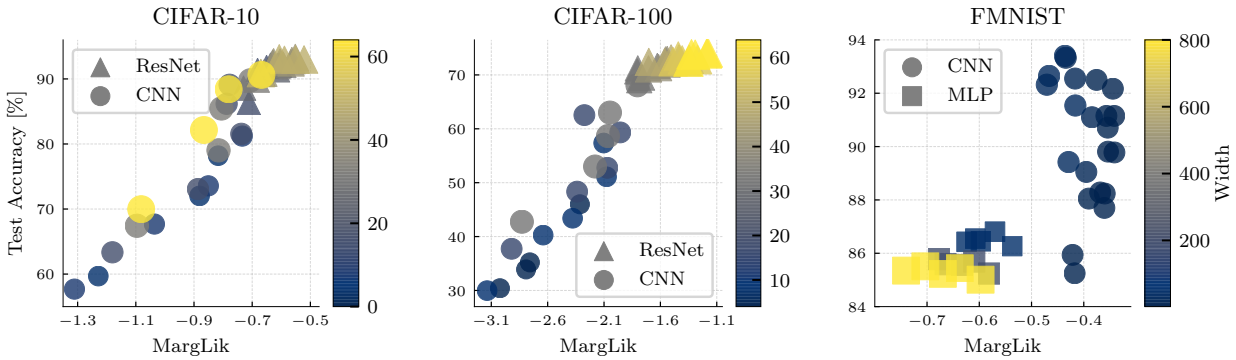


Figure C.3: Figure corresponds to Fig. 2 but shows markers colored and scaled by **width**.

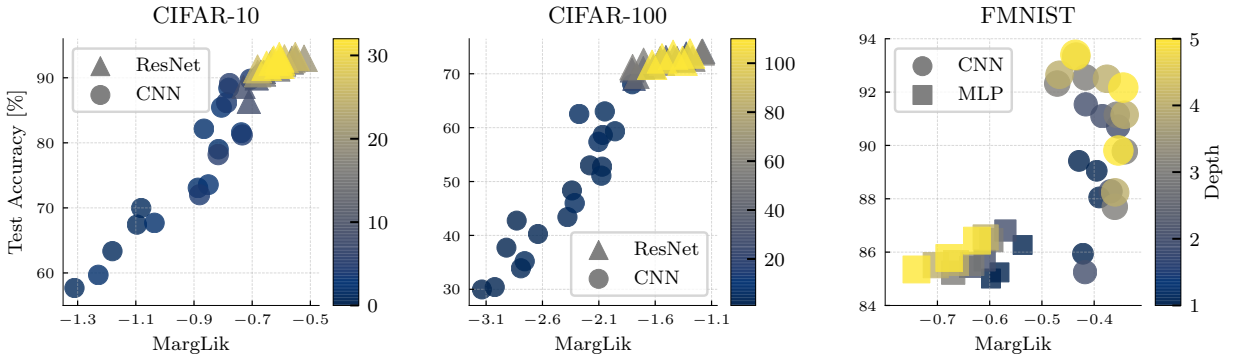


Figure C.4: Figure corresponds to Fig. 2 but shows markers colored and scaled by **depth**.

accuracy	MargLik	width	depth	# params	model
85.01	−0.598	800	1	636,034	MLP
85.23	−0.670	800	3	1,917,650	MLP
85.24	−0.581	200	1	159,034	MLP
85.25	−0.417	2	2	566	CNN
85.35	−0.740	800	5	3,199,266	MLP
85.44	−0.637	800	2	1,276,842	MLP
85.51	−0.703	800	4	2,558,458	MLP
85.56	−0.633	200	3	239,450	MLP
85.59	−0.664	200	4	279,658	MLP
85.69	−0.610	200	2	199,242	MLP
85.78	−0.677	200	5	319,866	MLP
85.94	−0.421	4	1	748	CNN
86.27	−0.537	50	1	39,784	MLP
86.43	−0.624	50	5	50,016	MLP
86.45	−0.597	50	3	44,900	MLP
86.55	−0.608	50	4	47,458	MLP
86.81	−0.570	50	2	42,342	MLP
87.70	−0.361	2	3	864	CNN
88.05	−0.390	8	1	1428	CNN
88.24	−0.360	2	4	2074	CNN
88.28	−0.369	4	2	1166	CNN
89.05	−0.395	16	1	2788	CNN
89.42	−0.429	32	1	5508	CNN
89.78	−0.341	4	3	2296	CNN
89.81	−0.354	2	5	6756	CNN
90.71	−0.354	8	2	2798	CNN
91.10	−0.384	16	2	7790	CNN
91.15	−0.342	4	4	6978	CNN
91.15	−0.357	8	3	7320	CNN
91.54	−0.416	32	2	24,686	CNN
92.17	−0.344	4	5	25,516	CNN
92.32	−0.471	32	3	97,944	CNN
92.50	−0.376	8	4	25,858	CNN
92.55	−0.417	16	3	26,008	CNN
92.65	−0.466	32	4	393,154	CNN
93.33	−0.434	16	4	99,906	CNN
93.40	−0.436	32	5	1,573,356	CNN

Table C11: FMNIST architectures

accuracy	MargLik	width	depth	# params	model
57.69	−1.311	8	1	1572	CNN
59.69	−1.229	4	2	1238	CNN
63.35	−1.180	16	1	3076	CNN
67.43	−1.096	32	1	6084	CNN
67.69	−1.036	8	2	2942	CNN
69.98	−1.081	64	1	12,100	CNN
72.02	−0.881	4	4	7050	CNN
73.07	−0.886	16	2	8078	CNN
73.57	−0.851	8	3	7464	CNN
78.20	−0.817	4	5	25,588	CNN
79.02	−0.816	32	2	25,262	CNN
81.21	−0.734	8	4	26,002	CNN
81.56	−0.737	16	3	26,296	CNN
82.15	−0.866	64	2	87,278	CNN
85.46	−0.806	32	3	98,520	CNN
86.18	−0.788	16	4	100,194	CNN
86.30	−0.712	16	8	75,129	$ResNet$
88.41	−0.781	64	3	381,208	CNN
88.58	−0.730	24	8	167,825	$ResNet$
89.10	−0.778	16	5	395,404	CNN
89.78	−0.706	32	4	393,730	CNN
89.90	−0.685	32	8	297,385	$ResNet$
90.62	−0.662	16	14	172,128	$ResNet$
90.62	−0.668	64	4	1,561,410	CNN
90.63	−0.681	16	20	269,127	$ResNet$
90.89	−0.660	40	8	463,809	$ResNet$
90.90	−0.670	48	8	667,097	$ResNet$
91.37	−0.682	16	26	366,126	$ResNet$
91.40	−0.613	24	14	385,784	$ResNet$
91.83	−0.651	16	32	463,125	$ResNet$
91.97	−0.613	32	20	1,071,991	$ResNet$
92.09	−0.610	24	26	821,702	$ResNet$
92.11	−0.637	24	20	603,743	$ResNet$
92.12	−0.606	24	32	1,039,661	$ResNet$
92.17	−0.593	32	14	684,688	$ResNet$
92.47	−0.567	40	14	1,068,840	$ResNet$
92.50	−0.630	32	26	1,459,294	$ResNet$
92.62	−0.626	40	32	2,883,933	$ResNet$
92.66	−0.604	32	32	1,846,597	$ResNet$
92.81	−0.564	40	20	1,673,871	$ResNet$
92.93	−0.589	48	26	3,280,526	$ResNet$
92.98	−0.551	48	14	1,538,240	$ResNet$
92.99	−0.522	48	20	2,409,383	$ResNet$
93.16	−0.607	48	32	4,151,669	$ResNet$
93.29	−0.552	40	26	2,278,902	$ResNet$

Table C9: CIFAR-10 architectures

accuracy	MargLik	width	depth	# params	model
23.54	−3.435	4	1	6670	CNN
29.97	−3.137	8	1	13,182	CNN
30.41	−3.023	4	2	7808	CNN
33.90	−2.791	4	3	8218	CNN
35.21	−2.758	4	4	12,900	CNN
37.73	−2.920	16	1	26,206	CNN
40.27	−2.640	8	2	15,992	CNN
42.75	−2.829	32	1	52,254	CNN
43.40	−2.379	8	3	19,074	CNN
45.99	−2.314	4	5	31,438	CNN
48.34	−2.339	16	2	34,088	CNN
51.09	−2.078	8	4	37,612	CNN
52.76	−2.071	16	3	49,426	CNN
53.00	−2.180	32	2	77,192	CNN
57.36	−2.103	8	5	111,510	CNN
58.65	−2.065	32	3	144,690	CNN
59.33	−1.956	16	4	123,324	CNN
62.54	−2.274	16	5	418,534	CNN
63.01	−2.047	32	4	439,900	CNN
68.07	−1.804	32	5	1,620,102	CNN
69.01	−1.788	32	20	1,083,601	$ResNet$
69.50	−1.763	32	44	2,632,813	$ResNet$
70.65	−1.805	32	32	1,858,207	$ResNet$
71.11	−1.706	40	20	1,688,361	$ResNet$
71.22	−1.801	32	56	3,407,419	$ResNet$
71.35	−1.616	40	32	2,898,423	$ResNet$
71.66	−1.627	32	110	6,893,146	$ResNet$
72.02	−1.540	40	56	5,318,547	$ResNet$
72.36	−1.562	48	20	2,426,753	$ResNet$
72.38	−1.358	48	110	15,493,898	$ResNet$
72.40	−1.703	48	56	7,653,611	$ResNet$
72.45	−1.487	56	20	3,298,777	$ResNet$
72.47	−1.351	64	32	7,401,471	$ResNet$
72.78	−1.554	40	44	4,108,485	$ResNet$
72.78	−1.407	56	44	8,041,333	$ResNet$
72.82	−1.271	56	56	10,412,611	$ResNet$
72.85	−1.540	40	110	10,763,826	$ResNet$
72.90	−1.344	56	32	5,670,055	$ResNet$
73.02	−1.436	48	44	5,911,325	$ResNet$
73.05	−1.505	48	32	4,169,039	$ResNet$
73.51	−1.441	56	110	21,083,362	$ResNet$
73.75	−1.291	64	110	27,532,218	$ResNet$
73.82	−1.325	64	20	4,304,433	$ResNet$
73.98	−1.188	64	56	13,595,547	$ResNet$
74.16	−1.180	64	44	10,498,509	$ResNet$

Table C10: CIFAR-100 architectures

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.