Stochastic Marginal Likelihood Gradients using Neural Tangent Kernels

Alexander Immer [1 2] **Tycho F.A. van der Ouderaa** [3] **Mark van der Wilk** [3]
Gunnar Rätsch [1] **Bernhard Schölkopf** [2]

Abstract

Selecting hyperparameters in deep learning greatly impacts its effectiveness but requires manual effort and expertise. Recent works show that Bayesian model selection with Laplace approximations can allow to optimize such hyperparameters just like standard neural network parameters using gradients and on the training data. However, estimating a single hyperparameter gradient requires a pass through the entire dataset, limiting the scalability of such algorithms. In this work, we overcome this issue by introducing lower bounds to the linearized Laplace approximation of the marginal likelihood. In contrast to previous estimators, these bounds are amenable to stochastic-gradient-based optimization and allow to trade off estimation accuracy against computational complexity. We derive them using the function-space form of the linearized Laplace, which can be estimated using the neural tangent kernel. Experimentally, we show that the estimators can significantly accelerate gradient-based hyperparameter optimization.

1. Introduction

Deep learning models have many hyperparameters that need to be chosen properly to achieve good performance. For example, making the right architecture, regularization, or data augmentation choices remains a problem that requires significant human effort and computation. This is even apparent within the field of deep learning itself, where the state-of-the-art settings change frequently. Such manual trial-and-error procedure is inefficient and can hinder the application of deep learning to new problems. A more automatic procedure could be much more sustainable.

[1]Department of Computer Science, ETH Zurich, Switzerland
[2]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[3]Imperial College London, UK. Correspondence to: Alexander Immer <alexander.immer@inf.ethz.ch>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

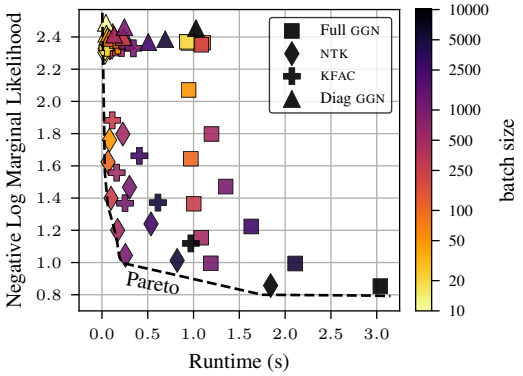

Figure 1. Pareto frontier between marginal likelihood estimator tightness and runtime for invariance learning on rotated MNIST. Our proposed estimators enable stochastic approximation with varying batch size in contrast to the four previously used full-batch estimators (batch size of 10 000 in black). NTK-based estimators on subsets of data perform well at more than 10× speed-up.

Automated machine learing (AutoML) methods have the potential to greatly reduce the cost of deploying deep learning for new problems (Hutter et al., 2019). Commonly, black-box algorithms are applied in this setting to optimize hyperparameters towards the best possible validation performance. For example, neural architectures can be optimized by training each option to convergence and assessing the validation performance repeatedly. However, such iterative procedures require training many models to convergence and suffer from high-dimensional hyperparameters due to their black-box nature. We posit that (stochastic-)gradient-based optimization of hyperparameters would be more desirable.

Bayesian model selection, where we consider hyperparameters and neural network weights jointly as part of a probabilistic model, is amenable to gradient-based optimization and also does not require any validation data (MacKay, 2003). Gradient-based optimization of hyperparameters tends to suffer less from high dimensionality than iterative black-box methods (Lorraine et al., 2020). Further, in such integrated procedure, the hyperparameters can be jointly optimized with the neural network weights (Foresee & Hagan, 1997; Immer et al., 2021a). While this is theoretically appealing, estimating the hyperparameter gradient is costly as it requires differentiating the *marginal likelihood*, i.e., the normalization constant of the posterior.

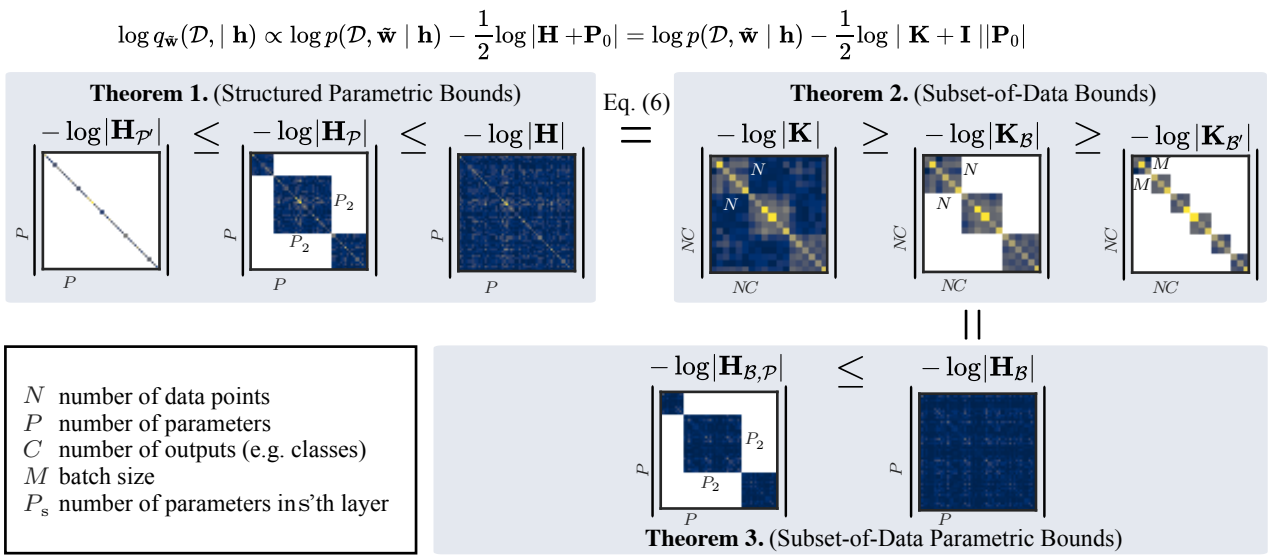

Figure 2. Schematic overview of the derived lower bounds to the linearized Laplace marginal likelihood. The parametric bounds on the top left justify commonly used block-diagonal and diagonal Hessian and Gauss-Newton approximations. The bounds on the right are derived from the dual NTK perspective and give a novel family of lower bounds. These lower bounds enable stochastic marginal likelihood estimation and gradients with both NTK-based estimators and parametric variants, e.g., Gauss-Newton, by translating them back (bottom).

To enable Bayesian model selection of hyperparameters in deep learning with marginal likelihood gradients, Immer et al. (2021a) recently revisited the use of the Laplace approximation (MacKay, 1992a) with modern Gauss-Newton approximations. While these structured linearized Laplace approximations can be used for architecture comparison and gradient-based learning of data augmentations and regularization strength (Immer et al., 2021a; 2022a;b; Antorán et al., 2022b; Daxberger et al., 2021; Hataya & Nakayama, 2021), each hyperparameter gradient requires a pass through the entire training dataset. In particular, the objective does not allow (unbiased) stochastic gradient estimation.

In the present work, we derive lower bounds to the Laplace approximation of the marginal likelihood that are amenable to stochastic-gradient-based optimization and show that currently used structured approximations are in fact lower bounds (overview in Figure 2). Instead of a whole pass through the training data, our estimators require only batches of data to estimate hyperparameter gradients, which can greatly increase the scalability of such algorithms. The size of the batches can further be chosen to achieve a trade-off between estimation quality and runtime complexity (cf. Figure 1). We derive the lower bounds from the dual form of the linearized Laplace (Khan et al., 2019; Immer et al., 2021b), which relies on the (empirical) neural tangent kernel (NTK, Jacot et al., 2018). By partitioning the NTK using batches, we obtain a lower bound to the linearized Laplace marginal likelihood, which permits unbiased stochastic estimates and gradients. Empirically, our estimators enable up to 25-fold acceleration of hyperparameter optimization and application to larger datasets.

2. Background

We consider supervised learning tasks with inputs $\mathbf{x}_n \in \mathbb{R}^D$ and targets $\mathbf{y}_n \in \mathbb{R}^C$ giving dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ with N pairs. We model the targets given inputs with a neural network. We parameterize the neural network $\mathbf{f}$, as usual, with weights $\mathbf{w} \in \mathbb{R}^P$, but additionally make the dependence on eventual hyperparameters $\mathbf{h}$, such as architecture, explicit, and have $\mathbf{f}(\mathbf{x}; \mathbf{w}, \mathbf{h}) \in \mathbb{R}^C$. Assuming the data are *i.i.d.*, we can define a probabilistic model with likelihood $p(\mathcal{D}|\mathbf{w}, \mathbf{h}) = \prod_{n=1}^N p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{w}, \mathbf{h})$ that depends on the neural network. Further, we have a prior $p(\mathbf{w}|\mathbf{h})$.

2.1. Bayesian Model Selection

In principle, Bayesian inference makes no distinction between hyperparameters $\mathbf{h}$ and weights $\mathbf{w}$ and simply infers both jointly. With an additional prior $p(\mathbf{h})$, this means that we obtain the joint posterior $p(\mathbf{w}, \mathbf{h}|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{w}, \mathbf{h})p(\mathbf{w}|\mathbf{h})p(\mathbf{h})$. While there exist methods to approximate the posterior $p(\mathbf{w}|\mathcal{D}, \mathbf{h})$ efficiently in deep learning, Bayesian inference of the hyperparameters $\mathbf{h}$ is complicated due to their heterogeneous and complicated structure, e.g., it could require a distribution over neural architectures. Instead, it is common to perform *empirical Bayes* to select hyperparameters $\mathbf{h}$ by optimizing the *marginal likelihood*,

$$p(\mathcal{D}|\mathbf{h}) = \int p(\mathcal{D}|\mathbf{w}, \mathbf{h})p(\mathbf{w}|\mathbf{h})\,\mathrm{d}\mathbf{w}\,, \tag{1}$$

which implements Occam's razor, performing a trade-off between model complexity and accuracy (Rasmussen & Ghahramani, 2001; Bishop, 2006). This point approxima-

tion can work well unless the number of hyperparameters is too large, which leads to overfitting (Ober et al., 2021).

We can optimize the marginal likelihood by gradient ascent on the differentiable hyperparameters,

$$\mathbf{h}_{t+1} \leftarrow \mathbf{h}_t + \nabla_{\mathbf{h}} \log p(\mathcal{D}|\mathbf{h})|_{\mathbf{h}=\mathbf{h}_t}, \tag{2}$$

similarly to how we differentiate the log joint, or regularized log likelihood, to optimize neural network weights $\mathbf{w}$.

2.2. Linearized Laplace Approximation

The linearized Laplace (LA) approximation (MacKay, 1992a) provides an approximation to the log marginal likelihood and its scalable variants are currently among the few approximations that have proven effective for deep learning hyperparameter optimization. First, the neural network is linearized at a mode of the posterior $\tilde{\mathbf{w}}$,

$$\mathbf{f}^{\text{lin}}_{\tilde{\mathbf{w}}}(\mathbf{x}; \mathbf{w}, \mathbf{h}) \stackrel{\text{def}}{=} \mathbf{f}(\mathbf{x}; \tilde{\mathbf{w}}, \mathbf{h}) + \mathbf{J}_{\tilde{\mathbf{w}}}(\mathbf{x}; \mathbf{h})(\mathbf{w} - \tilde{\mathbf{w}}), \tag{3}$$

where $\mathbf{J}$ denotes the Jacobian of the neural network w.r.t. weights with entries $[\mathbf{J}_{\tilde{\mathbf{w}}}]_{cp} = \frac{\partial f_c(\mathbf{x};\mathbf{w},\mathbf{h})}{\partial w_p}|_{\mathbf{w}=\tilde{\mathbf{w}}}$. Further, we have the Hessian of the negative log likelihood w.r.t. $\mathbf{f}$,

$$\mathbf{\Lambda}_{\tilde{\mathbf{w}}}(\mathbf{x}; \mathbf{h}) \stackrel{\text{def}}{=} -\nabla^2_{\mathbf{f}} \log p(\mathbf{y}|\mathbf{x}, \tilde{\mathbf{w}}, \mathbf{h}) \in \mathbb{R}^{C \times C}, \tag{4}$$

which is positive semidefinite for common likelihoods used in classification and regression (Murphy, 2012).

The linearized Laplace approximation $\log q(\mathcal{D}|\mathbf{h})$ to the log marginal likelihood $\log p(\mathcal{D}|\mathbf{h})$ is then given by

$$\begin{aligned} \log q_{\tilde{\mathbf{w}}}(\mathcal{D}|\mathbf{h}) = \log p(\mathcal{D}|\tilde{\mathbf{w}}, \mathbf{h}) + \log p(\tilde{\mathbf{w}}|\mathbf{h}) \\ - \tfrac{1}{2} \log |\mathbf{J}^\intercal_{\tilde{\mathbf{w}}} \mathbf{\Lambda}_{\tilde{\mathbf{w}}} \mathbf{J}_{\tilde{\mathbf{w}}} + \mathbf{P}_0| + \tfrac{P}{2} \log 2\pi, \end{aligned} \tag{5}$$

where $\mathbf{J}_{\tilde{\mathbf{w}}} \in \mathbb{R}^{NC \times P}$ denotes stacked Jacobians $\mathbf{J}_{\tilde{\mathbf{w}}}(\mathbf{x}_n; \mathbf{h})$, and $\mathbf{\Lambda}_{\tilde{\mathbf{w}}} \in \mathbb{R}^{NC \times NC}$ is a block-diagonal matrix with blocks $\mathbf{\Lambda}_{\tilde{\mathbf{w}}}(\mathbf{x}_n; \mathbf{h})$ for $n = 1, .., N$, respectively, and we have suppressed the dependency on hyperparameters $\mathbf{h}$ for notational brevity. The matrix $\mathbf{P}_0 \in \mathbb{R}^{P \times P}$ is the Hessian w.r.t. weights $\mathbf{w}$ of the log prior $\log p(\mathbf{w}|\mathbf{h})$ and is diagonal here. Because the first two terms of the approximation constitute the log joint, i.e., the training loss, all terms are simple to estimate and differentiate except for the (log-)determinant, which can be written equivalently (Immer et al., 2021a) as

$$|\mathbf{J}^\intercal_{\tilde{\mathbf{w}}} \mathbf{\Lambda}_{\tilde{\mathbf{w}}} \mathbf{J}_{\tilde{\mathbf{w}}} + \mathbf{P}_0| = |\mathbf{J}_{\tilde{\mathbf{w}}} \mathbf{P}_0^{-1} \mathbf{J}^\intercal_{\tilde{\mathbf{w}}} \mathbf{\Lambda}_{\tilde{\mathbf{w}}} + \mathbf{I}||\mathbf{P}_0|, \tag{6}$$

and allows us to either compute the log-determinant of a $P \times P$ or $NC \times NC$ matrix. Unfortunately, the log-determinant prohibits unbiased stochastic gradients on batches of data and is therefore not amenable to optimization with SGD.

2.3. Generalized Gauss-Newton and Neural Tangents

The two alternative ways to compute the determinant in Equation 6 use either the generalized Gauss-Newton (GGN) form or the neural tangent kernel (NTK) form. In particular,

$$\mathbf{H} \stackrel{\text{def}}{=} \mathbf{J}^\intercal_{\tilde{\mathbf{w}}} \mathbf{\Lambda}_{\tilde{\mathbf{w}}} \mathbf{J}_{\tilde{\mathbf{w}}} \quad \text{and} \quad \mathbf{K} \stackrel{\text{def}}{=} \mathbf{J}_{\tilde{\mathbf{w}}} \mathbf{P}_0^{-1} \mathbf{J}^\intercal_{\tilde{\mathbf{w}}} \mathbf{\Lambda}_{\tilde{\mathbf{w}}} \tag{7}$$

are the GGN approximation to the Hessian and the NTK with additional scaling by $\mathbf{P}_0^{-1}$ and $\mathbf{\Lambda}_{\tilde{\mathbf{w}}}$, respectively.[1] To obtain marginal likelihood gradients, we need to estimate and differentiate these two quantities. However, they are intractable for deep neural networks due to large numbers of parameters P and many data points and outputs NC.

The GGN can be made tractable by structured block-diagonal and diagonal approximations (Martens & Grosse, 2015). Without loss of generality, we assume ordered parameter indices $\{1, .., P\}$. With a contiguous partition[2] $\mathcal{P} = \{\mathcal{P}_s\}_{s=1}^S$ of parameter indices, we have

$$\mathbf{H} \approx \mathbf{H}_{\mathcal{P}} = \begin{bmatrix} \mathbf{H}_{\mathcal{P}_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{H}_{\mathcal{P}_S} \end{bmatrix} = \mathbf{H}_{\mathcal{P}_1} \underbrace{\oplus ... \oplus}_{\text{block-diagonal stacking}} \mathbf{H}_{\mathcal{P}_S}, \tag{8}$$

a block-diagonal approximation where each block corresponds to a subset $\mathcal{P}_s$. For example, if each $\mathcal{P}_s$ corresponds to the parameters of one particular layer, we recover a block-diagonal layer-wise approximation like KFAC (Martens & Grosse, 2015). If each set only contains a single parameter index, $\mathcal{P}_s = \{s\}$, we recover a diagonal approximation.

2.4. Hyperparameter Optimization for Deep Learning

To optimize general hyperparameters, we estimate and differentiate Laplace approximations during neural network training using scalable approximations (Immer et al., 2021a). The form of hyperparameters $\mathbf{h}$ governs the complexity of obtaining their gradients. For example, the typical problem of optimizing the prior precision, $\mathbf{P}_0$, is relatively simple because it acts linearly on the GGN and NTK. However, general hyperparameters, such as invariances, affect the forward pass of the neural network and act non-linearly on GGN and NTK. Therefore, they require higher-order differentiation.

In the following, we illustrate the derived bounds for both settings, i.e., selecting a scalar or layer-wise prior precision (weight decay) on MNIST, and rotational invariance on rotated MNIST. We use a subset of 1000 random digits and apply a small 3-layer convolutional neural network with linear output layer. This setting ensures that we can compute the exact linearized Laplace in Equation 5 as a reference.

[1] We recover the standard finite width NTK for a Gaussian likelihood and prior with unit variance, i.e., $\mathbf{K} = \mathbf{J}_{\tilde{\mathbf{w}}} \mathbf{J}^\intercal_{\tilde{\mathbf{w}}}$.

[2] A partition of a set $\mathcal{S}$ is a set $\mathcal{P}$ of disjoint subsets of $\mathcal{S}$ whose union is equal to $\mathcal{S}$. It is contiguous if the elements in $\mathcal{P}$ are ordered and the order remains preserved in $\mathcal{S}$.

3. Structured Parametric Laplace Approximations are Lower Bounds

Practical linearized Laplace approximations of deep neural networks use structured GGN approximations, such as diagonal and block-diagonal variants, to make the log-determinant computation tractable (Immer et al., 2021a; Daxberger et al., 2021; Antorán et al., 2022b). However, it is unclear to what extent these approximations are related to the exact linearized LA. The following Theorem shows that such approximations are in fact lower bounds and therefore justifies their use better. This is similar to variational inference (Blei et al., 2017), where more restrictive approximations, such as a diagonal instead of a multivariate Gaussian, lead to more slack in the evidence lower bound.

Theorem 1 (Parametric structured lower bound)**.** *For any partitioning $\mathcal{P}$ of the parameter indices $\{1, .., P\}$, the block-diagonal approximation of $\mathbf{H}_\mathcal{P}$ to $\mathbf{H}$ results in a lower bound to the* LA *log marginal likelihood (Equation 5), i.e.,*

$$\log q_{\tilde{\mathbf{w}}}(\mathcal{D}|\mathbf{h}) \geq \log p(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) - \tfrac{1}{2} \log |\mathbf{H}_\mathcal{P} + \mathbf{P}_0| + c, \quad (9)$$

where $c = \frac{P}{2} \log 2\pi$. Further, a refinement[3] of the partition $\mathcal{P}$ to $\mathcal{P}'$ will result in a lower bound with even more slack.

Example 1.1. *The block-diagonal Laplace approximation that only considers correlations within layers, similar to the popular* KFAC[4]*-Laplace (Ritter et al., 2018; Daxberger et al., 2021), is a lower bound to the linearized Laplace approximation. A diagonal Laplace approximation that only considers marginal variances of parameters is a further lower bound of such block-diagonal approximation.*

For a proof, refer to App. A. The bound holds for commonly used block-diagonal approximations and is further agnostic to the type of Hessian approximation. That is, it holds for the Hessian, GGN, or (empirical) Fisher variants of the LA alike. In Figure 3, we show the bound for different hyperparameter values $\mathbf{h}$ of the prior precision and rotational invariance in the illustrative setting described in Sec. 2.4. For the prior precision, only the diagonal approximation has a large slack and significantly different optimum, which is in line with previous empirical observations (MacKay, 1995; Daxberger et al., 2021). Perhaps surprisingly, it can still be useful to select the right rotational invariance because its shape and thus optimum is in line with the better approximations.

While the above bounds justify existing Laplace approximations, they do not improve scalability with dataset size, because we have partitioned the parameters rather than the dataset. To do so, we employ the dual NTK representation.

[3]A refinement $\mathcal{P}'$ of $\mathcal{P}$ is a partition such that for all $\mathcal{P}'_s \in \mathcal{P}'$ there exists $\mathcal{P}_s \in \mathcal{P}$ such that $\mathcal{P}'_s \subseteq \mathcal{P}_s$.

[4]Note that KFAC is a further approximation to the block-diagonal and a bound is not theoretically guaranteed. Nonetheless, it does seem to hold in practice.

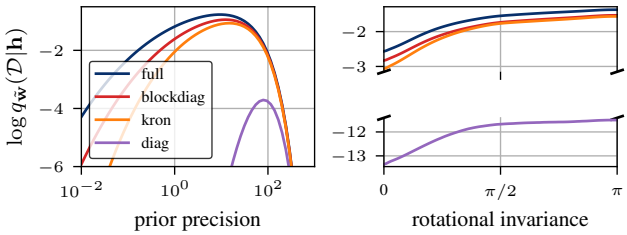

Figure 3. Illustration of the parametric bound (Theorem 1) with commonly used approximation structures for prior precision and rotational invariance as hyperparameters $\mathbf{h}$ on MNIST with CNN.

4. Stochastic Gradients using the NTK

While structured parametric lower bounds to the linearized Laplace marginal likelihood give good performance on model selection, a single hyperparameter gradient still requires an entire pass through the training data. This is because the log-determinant is in general not separable across data points and computing it on a subset of data would lead to an uncontrolled upper bound to the marginal likelihood. To overcome this limitation, we use the log-determinant in its dual NTK form in Equation 6 to devise lower bounds that enable batched computation on subsets of data and thus stochastic-gradient-based optimization. Our bounds work for NTK (Sec. 4.1) or parametric and structured GGN (Sec. 4.2) estimators alike and enable tightening the lower bound using larger batches of data or improving the partitioning of data points into batches (Sec. 4.3).

4.1. Subset-of-Data Kernel Bound

Using the NTK form of the log-determinant, we construct lower bounds for subsets of data that enable stochastic marginal likelihood gradients and can therefore greatly improve the efficiency of hyperparameter optimization. Due to the cubic scaling with the number of data points, the NTK form has so far been only used in special cases (Immer et al., 2021a; Antorán et al., 2022a). Using our bounds, however, the NTK form on subsets of data becomes a tractable alternative for parametric variants like KFAC. Further, computing the NTK can be more architecture-agnostic (Novak et al., 2019; 2022) because it relies on plain Jacobians while structured approximations like KFAC are often non-trivial to compute or extend to custom architectures (Dangel et al., 2019; Osawa, 2021; Botev & Martens, 2022).

Instead of partitioning the parameters as in Theorem 1, we partition the N inputs and C outputs and make use of the NTK form, which naively requires computing a $NC \times NC$ matrix, to obtain a lower bound for subsets of data. In particular, we have the set of indices $\mathcal{I} = \{(n, c) \mid 1 \leq n \leq N, 1 \leq c \leq C\}$ corresponding to inputs $\mathbf{x}_n$ and outputs f_c of the neural network. The index order is fixed w.r.t. $\mathbf{K}$ but arbitrary. With partitioning $\mathcal{B} = \{\mathcal{B}_m\}_{m=1}^M$ of $\mathcal{I}$, we have

$$\mathbf{K} \approx \mathbf{K}_\mathcal{B} = \mathbf{K}_{\mathcal{B}_1} \oplus \cdots \oplus \mathbf{K}_{\mathcal{B}_M}, \quad (10)$$

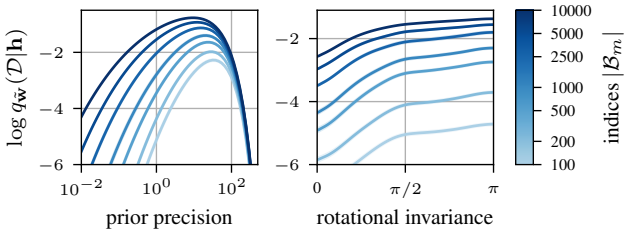

Figure 4. Illustration of the subset-of-data NTK lower bound (Theorem 2) with varying subset sizes for prior precision and rotational invariance as hyperparameters $\mathbf{h}$ on the two MNIST variants.

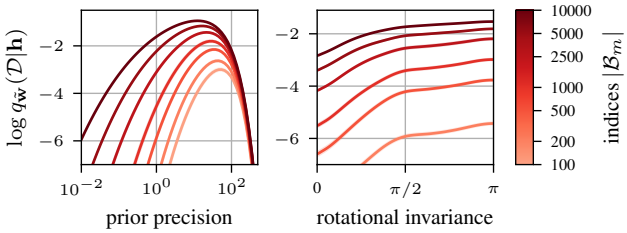

Figure 5. Illustration of the parametric doubly lower bound using the block-diagonal GGN (Corollary 3.1) with varying subset sizes for prior and invariance hyperparameters. Setup as in Figure 4.

where each $\mathbf{K}_{\mathcal{B}_m}$ is a NTK on a subset of data and outputs. Mathematically, we have $\mathbf{K}_{\mathcal{B}_m} = \mathbf{J}_{\mathcal{B}_m}\mathbf{P}_0^{-1}\mathbf{J}_{\mathcal{B}_m}^\intercal \mathbf{\Lambda}_{\mathcal{B}_m}$ where $\mathbf{J}_{\mathcal{B}_m} \in \mathbb{R}^{|\mathcal{B}_m|\times P}$ are the Jacobians for the input-output pairs and $\mathbf{\Lambda}_{\mathcal{B}_m} \in \mathbb{R}^{|\mathcal{B}_m|\times|\mathcal{B}_m|}$ the corresponding log-likelihood Hessian terms.

Theorem 2 (Data subset lower bound)**.** *For any partitioning $\mathcal{B}$ of inputs and outputs $\mathcal{I}$, the corresponding block-diagonal* NTK *approximation $\mathbf{K}_\mathcal{B}$ to $\mathbf{K}$ results in a lower bound to the linearized Laplace marginal likelihood, i.e.,*

$$\begin{aligned}\log q_{\tilde{\mathbf{w}}}(\mathcal{D}|\mathbf{h}) &\geq \log p(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) - \tfrac{1}{2}\log|\mathbf{K}_\mathcal{B} + \mathbf{I}||\mathbf{P}_0| + c \\ &\propto \log p(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) - \tfrac{1}{2}\log|\mathbf{P}_0| - \tfrac{1}{2}\textstyle\sum_{m=1}^M \log|\mathbf{K}_{\mathcal{B}_m} + \mathbf{I}|,\end{aligned} \tag{11}$$

where $c = \frac{P}{2}\log 2\pi$ is left out in the second line indicated by the proportionality. Further, a refinement of partition $\mathcal{B}$ to $\mathcal{B}'$ will result in a lower bound with more slack.

We give a proof in App. A. Theorem 2 shows that we can use the NTK on subsets of data and outputs (for multi-output cases like classification) and obtain a valid lower bound. Further, the form of Equation 11 allows unbiased stochastic estimation, gradients, and therefore SGD-based optimization of hyperparameters using (mini-)batches of data. That is, by uniformly sampling index sets $\mathcal{B}_m$, we have

$$\textstyle\sum_{m=1}^M \log|\mathbf{K}_{\mathcal{B}_m} + \mathbf{I}| = M\mathbb{E}_{m\sim\mathcal{U}[M]}[\log|\mathbf{K}_{\mathcal{B}_m} + \mathbf{I}|]. \tag{12}$$

In Figure 4, we show the tightness of the bound for different subset sizes $|\mathcal{B}_m|$ on the two illustrative problems. Despite small subset sizes, the bounds are tighter than the crude diagonal approximation and already around 2% of the input-output pairs can suffice to select good hyperparameters.

4.2. Subset-of-Data Parametric Bounds

Computing the NTK on subsets of data can be efficient but is non-trivial to parallelize, in which case parametric approximations like KFAC might be preferable (Osawa et al., 2019). Using the NTK lower bound in Theorem 2, we can devise an equivalent parametric estimator, which gives a lower bound on subsets of data (and parameters). This enables the use of approximations like KFAC with Laplace on batches of data and thus for SGD-based hyperparameter optimization.

Defining $\mathbf{H}_{\mathcal{B}_m} \stackrel{\text{def}}{=} \mathbf{H}_{\mathcal{B}_m} = \mathbf{J}_{\mathcal{B}_m}^\intercal \mathbf{\Lambda}_{\mathcal{B}_m}\mathbf{J}_{\mathcal{B}_m}$ as a form of GGN on the subset of inputs and outputs $\mathcal{B}_m$, we have equivalence to the NTK bound, which we show in App. A:

Theorem 3 (Parametric data subset bound)**.** *The* NTK*-based lower bound in Equation 11 to the linearized Laplace marginal likelihood is equivalent to a parametric variant,*

$$\begin{aligned}&\log p(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) - \tfrac{1}{2}\log|\mathbf{P}_0| - \tfrac{1}{2}\textstyle\sum_{m=1}^M \log|\mathbf{K}_{\mathcal{B}_m} + \mathbf{I}| \\ &= \log p(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) + \tfrac{M\text{-}1}{2}\log|\mathbf{P}_0| \text{-} \tfrac{1}{2}\textstyle\sum_{m=1}^M \log|\mathbf{H}_{\mathcal{B}_m} + \mathbf{P}_0|,\end{aligned} \tag{13}$$

where $\mathbf{H}_{\mathcal{B}_m} = \mathbf{J}_{\mathcal{B}_m}^\intercal \mathbf{\Lambda}_{\mathcal{B}_m}\mathbf{J}_{\mathcal{B}_m}$ is estimated on a data subset.

This Theorem is useful because it shows how to do stochastic estimation of the marginal likelihood on a subset of data using a parametric form, which resembles the GGN, instead of the NTK. However, the full GGN is quadratic in the number of parameters and cannot be estimated in deep learning settings. To overcome this, we can further combine it with the parametric lower bound (Theorem 1) to justify structured parametric approximations like the block-diagonal and its scalable approximation, KFAC, on subsets of data.

Corollary 3.1 (Parametric doubly lower bound)**.** *For any partitioning $\mathcal{B}$ of input-output pairs $\mathcal{I}$ and $\mathcal{P}$ of parameter indices $\{1, .., P\}$, respectively, the following lower bound to the linearized Laplace marginal likelihood holds:*

$$\begin{aligned}\log q_{\tilde{\mathbf{w}}}(\mathcal{D}|\mathbf{h}) \geq &\log p(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) + \tfrac{M-1}{2}\log|\mathbf{P}_0| \\ &- \tfrac{1}{2}\textstyle\sum_{m=1}^M \log|\mathbf{H}_{\mathcal{B}_m,\mathcal{P}} + \mathbf{P}_0| + \tfrac{P}{2}\log 2\pi,\end{aligned} \tag{14}$$

where $\mathbf{H}_{\mathcal{B}_m,\mathcal{P}}$ is a block-structured GGN *approximation on subsets of input-output pairs.*

This doubly lower bound shows how parametric estimators, the most frequently used ones, can also be used for stochastic estimation and optimization of the log marginal likelihood. In Figure 5, we show the tightness of such bounds for the block-diagonal GGN. Surprisingly, even with just 5% of the input-output pairs, the bound is tighter then the diagonal approximation. Further, it only incurs a slight increase in slack compared to its NTK-based upper bound in Figure 4. In our experiments, we find that Corollary 3.1 successfully enables stochastic-gradient-based hyperparameter optimization with KFAC, which previously could only be used in the full-batch setting.

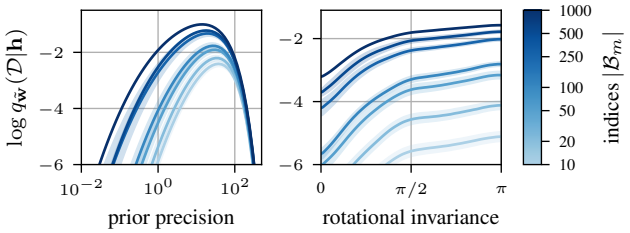

Figure 6. NTK Laplace bound using output-wise partitioning leads to tighter bounds and more efficiency. In comparison to Figure 4, the index set sizes are $C = 10$ times smaller corresponding to an improved complexity by a factor of C^3 at similar tightness.

4.3. Tighter Bounds with Better Partitions

The tightness of the lower bounds derived from the NTK variant can be controlled with the partitioning $\mathcal{B}$ into input-output pairs. $\mathcal{B}_m$ can be understood as batches like in parameter optimization using SGD. The bounds derived from the NTK motivate various choices of partitioning input data points and output dimensions so as to make the bounds as tight as possible. Mathematically, we want to find the partition $\mathcal{B}$ that maximizes the lower bound in Equation 11. We discuss two simple partitioning cases $\mathcal{B}$ that can make the bounds derived from the NTK potentially tighter: partitioning output dimensions and grouping inputs by labels.

Partitioning the NTK by outputs means that each partition $\mathcal{B}_m$ contains only one particular output and corresponds to an independent kernel as, for example, used in Gaussian processes (Rasmussen & Williams, 2006). Each $\mathbf{K}_{\mathcal{B}_m}$ is then an output-wise NTK, which can be much more efficient to compute than a full NTK. In the parametric space, this corresponds to a GGN for a single output and also greatly improves efficiency as it makes the GGN C times cheaper and is theoretically justified through a bound. Figure 6 shows that the output-wise NTK-based marginal likelihood bound is almost as tight as the full one at more than C times smaller cost. Already 0.2% of the input-output pairs, or equivalently 2% of the data, suffice to learn invariances. Across the entire design space of estimators, it is this class-wise partitioning that provides many Pareto-optimal estimators in Figure 1.

Alternatively, we can partition the index set $\mathcal{I}$ using the label information we have for each $\mathbf{x}_n$. In classification, we have access to labels y_n for each $\mathbf{x}_n$. Assuming that the correlation of the NTK intra-class is greater than the anticorrelation inter-class, it is reasonable to partition $\mathcal{I}$ in such a way that data points $\mathbf{x}_n$ with the same labels are in the same subset(s). In practice, this means that batches are made up from data points within the same class. We find that this approach can perform well but incurs a larger variance (cf. Figure 12 in App. C.1). In future work, it could be interesting to develop methods that track (anti-)correlation between inputs in the NTK to group them optimally during training. This is similar to inducing point optimization (Titsias, 2009).

Algorithm 1 Stochastic Marginal Likelihood Estimate

Input: dataset $\mathcal{D}$, likelihood, prior, random batch $\mathcal{B}_m \in \mathcal{B}$ (partition of input-output indices; $|\mathcal{B}| = M$), structure (e.g. NTK or GGN), param partition $\mathcal{P}$ (opt.).
Let $\mathcal{D}_m$ denote data points of input indices in $\mathcal{B}_m$
$\mathcal{L} \leftarrow \frac{|\mathcal{D}|}{|\mathcal{D}_m|} \log p(\mathcal{D}_m|\tilde{\mathbf{w}}, \mathbf{h}) + \log p(\tilde{\mathbf{w}}|\mathbf{h}) - \frac{1}{2} \log |\mathbf{P}_0|$
if structure = NTK **then**
  $\log q_{\tilde{\mathbf{w}}}(\mathcal{D}|\mathbf{h}) \leftarrow \mathcal{L} - \frac{M}{2} \log |\mathbf{K}_{\mathcal{B}_m} + \mathbf{I}|$
else
  $\log q_{\tilde{\mathbf{w}}}(\mathcal{D}|\mathbf{h}) \leftarrow \mathcal{L} - \frac{M}{2} \log |\mathbf{H}_{\mathcal{B}_m,\mathcal{P}} \mathbf{P}_0^{-1} + \mathbf{I}|$
end if
Return log marginal likelihood estimate $\log \tilde{q}\tilde{\mathbf{w}}(\mathcal{D}|\mathbf{h})$.

4.4. Family of Estimators and Algorithm

Through the NTK-based lower bound in Theorem 2, we derived estimators that enable stochastic marginal likelihood estimates and gradients. This allows us to optimize hyperparameters with SGD, just like neural network parameters. In particular, Equation 12 shows that we can obtain a stochastic unbiased estimator of the lower bounds on the log-determinants derived from the NTK. The remaining terms are the log likelihood, which allows for an unbiased stochastic estimate naively, and simple terms like the log determinant of the prior and constants.

In practice, we use our estimators for interleaved optimization of neural network weights and hyperparameters as in (Immer et al., 2021a). That is, we take gradient-steps on the hyperparameters every kth epoch after an initial burnin of b epochs. To obtain hyperparameter gradients, we choose a partitioning of the input-output indices $\mathcal{I}$ into $\mathcal{B}$, which contains M such batches, and then sample from these uniformly at random. Further, we resample the partition every epoch. In App. C.1, we show trajectories of this algorithm corresponding to the setting used for illustrating the bounds.

To keep the bounds tight, we recommend output-wise partitioning since it eliminates complexity scaling in the number of outputs C while empirically maintaining a relatively tight lower bound. In practice, the most scalable bounds for hyperparameter gradients are the NTK-based bounds and doubly bounds using an efficient parametric approximation like KFAC, which even work well with relatively small subset sizes. Interestingly, the output-wise approximation can also be applied to KFAC-GGN giving a differentiable and scalable alternative for general hyperparameter learning.

Alg. 1 shows how to compute stochastic marginal likelihood estimates. We use automatic differentiation to obtain gradients w.r.t. $\mathbf{h}$ to train them with SGD. To make the bound as tight as possible and thus improve the approximations, we choose the partitioning and the size of its batches so as to fully utilize the available memory in practice.

	Dataset	CIFAR-10				CIFAR-100			
$\mathbf{h}$	Estimator	$\log q_{\tilde{\mathbf{w}}}(\mathcal{D}\|\mathbf{h})$	log lik.	acc. [%]	time	$\log q_{\tilde{\mathbf{w}}}(\mathcal{D}\|\mathbf{h})$	log lik.	acc. [%]	time
prior precision	BASELINE	-	-1.22 ± 0.04	84.6 ± 0.3	35%	-	-3.35 ± 0.14	56.7 ± 1.1	8%
	NTK-500-1	-0.87 ± 0.02	-0.53 ± 0.01	86.8 ± 0.1	60%	**-1.98 ± 0.08**	-2.12 ± 0.04	61.7 + 0.6	16%
	KFAC-500-1	-1.41 ± 0.02	**-0.38 ± 0.00**	88.2 ± 0.1	65%	-4.73 ± 0.11	-1.31 ± 0.04	63.9 ± 0.9	24%
	KFAC-500-10	-1.31 ± 0.01	**-0.38 ± 0.00**	88.6 ± 0.1	81%	-3.95 ± 0.04	-1.26 ± 0.01	**67.9 ± 0.2**	70%
	KFAC-N-1	-1.10 ± 0.01	**-0.38 ± 0.00**	88.8 ± 0.1	52%	-4.31 ± 0.09	**-1.22 ± 0.02**	66.8 ± 0.5	15%
	KFAC-N-C	**-0.79 ± 0.01**	-0.39 ± 0.00	**89.2 ± 0.1**	100%	-2.38 ± 0.03	-1.65 ± 0.07	64.6 ± 0.4	100%
invariance	NTK-150-1	-0.56 ± 0.05	-0.42 ± 0.01	90.3 ± 0.2	9%	**-1.19 ± 0.07**	-1.45 ± 0.03	69.5 ± 0.8	4%
	KFAC-150-1	-0.59 ± 0.08	**-0.29 ± 0.00**	92.5 ± 0.2	10%	-2.22 ± 0.37	-1.40 ± 0.05	**71.8 ± 0.4**	5%
	KFAC-N-1	-0.44 ± 0.07	-0.31 ± 0.01	**93.0 ± 0.3**	23%	-3.43 ± 0.56	**-1.20 ± 0.03**	**71.7 ± 0.7**	10%
	KFAC-N-C	**-0.24 ± 0.02**	-0.33 ± 0.01	**93.2 ± 0.0**	100%	-	-	-	∼100%

Table 1. Benchmark of stochastic marginal likelihood estimators on CIFAR classification tasks with a Wide ResNet (16-4) learning layerwise prior precisions (top) and affine invariances (bottom). "500-1" refers to a batch size of 500 and single-output approximation with N and C corresponding to the full-batch setting. The timing is relative to the full-batch estimator, KFAC-N-C, which was previously used for these settings. Our stochastic estimators can perform on par at up to 25-fold speed-up. Due to the number of classes, KFAC-N-C runs out of memory for CIFAR-100. Performance bold per category if standard errors with best overlap from both sides.

5. Experiments

We experimentally validate the proposed estimators on various settings of marginal-likelihood-based hyperparameter optimization for deep learning. Overall, we find that the lower bounds using subsets of data or outputs often provide a better trade-off between performance and computational or memory complexity. In particular, they remain relatively tight even when applied only on small subsets of data and outputs. Therefore, they can greatly accelerate hyperparameter optimization with Laplace approximations, making marginal-likelihood optimization possible at larger scale.[5]

In our experiments, we optimize prior precision parameters, $\mathbf{P}_0$, equivalent to weight-decay, per layer of neural networks (Immer et al., 2021a; Antorán et al., 2022a) and learn invariances from data (van der Wilk et al., 2018; Immer et al., 2022b). Learning invariances requires differentiating the NTK or GGN and therefore acts as a realistic example of gradient-based optimization for general hyperparameters, which should be the long-term goal of Bayesian model selection for neural networks.

In the following, we first discuss the illustrative example used throughout the theoretical development. Further, we compare our estimators to the full dataset KFAC-Laplace approximation, which previously provided the best results for hyperparameter optimization with Laplace approximations (Immer et al., 2021a; Daxberger et al., 2021; Immer et al., 2022b). In this case, we find that estimators based on subsets of data are significantly more efficient and can perform on par even when the dataset is large. Lastly, we benchmark the two most scalable estimators on TinyImagenet, a setting, where full dataset Laplace approximations failed due to computational costs (Mlodozeniec et al., 2023).

[5]Code: `github.com/AlexImmer/ntk-marglik`

5.1. Tightness of Bounds, Performance, and Runtime

Throughout the theoretical development, we illustrate the tightness of the derived lower bounds to the linearized Laplace log marginal likelihood (cf. Figure 3, 4, 5, and 6). We use a small 3-layer convolutional network with linear output that has $P \approx 16\,000$ parameters on a random subset of 1000 MNIST (LeCun & Cortes, 2010) digits, which allows analytical computation of both the full GGN and NTK. For the illustration using the rotational invariance, we additionally rotate digits by a random angle $\theta \sim \mathcal{U}[-\pi, \pi]$. We compare the prior precision and rotational invariance hyperparameters with the log marginal likelihood estimate for the same trained neural network. In App. C.1, we additionally show the bound and corresponding test performance when optimizing it during training in both settings.

Parametric estimators. Figure 3 indicates that block-diagonal and KFAC variant attain almost the same marginal likelihood as a full GGN and have the same test performance. It also shows that the diagonal approximation can fail catastrophically due to its high slack in the bound (cf. Theorem 1) as it is the most refined partitioning $\mathcal{P}'$ of parameter indices.

Stochastic estimators. Figure 4 illustrates the proposed subset-of-data marginal likelihood estimators showing that already a small fraction of the input-output pairs leads to tight lower bounds. However, too small subset sizes can lead to failure when optimizing them, similar to the diagonal approximation. The doubly lower bound, which allows using structured parametric estimators on subsets of data (cf. Corollary 3.1), is displayed in Figure 5 and is similarly tight. Further, handling the $C = 10$ outputs independently for NTK-based estimators, which corresponds to a partitioning by class, almost performs as well as a full NTK in Figure 6 and is often significantly cheaper.

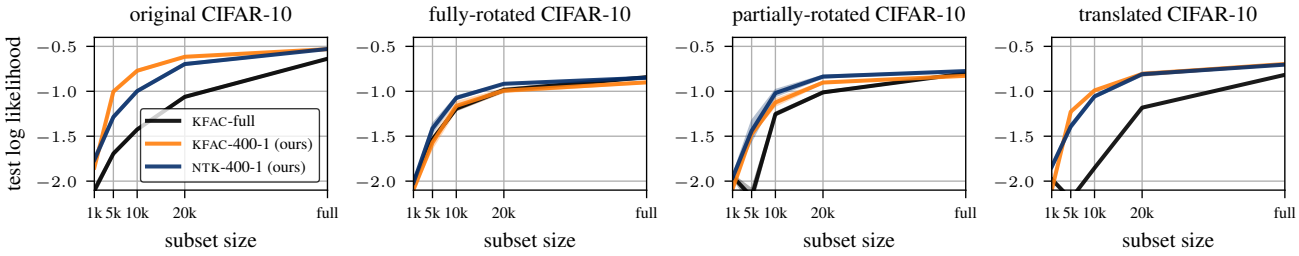

Figure 7. Proposed stochastic marginal likelihood estimators using NTK and KFAC with a fixed batch size of 400 and class-wise partitioning perform well on subsets and modifications of CIFAR-10 with a ResNet. Our estimators can take more stochastic gradient steps than the full KFAC estimator at a faster runtime and thus greatly improve over it in terms of test log likelihood. This suggests that using many stochastic gradients can be more effective for hyperparameter learning than taking few exact gradients. More results in App. C.4.

Pareto-efficiency. Figure 1 shows the obtained marginal likelihood estimates versus the runtime of a single estimate (and its gradient w.r.t. hyperparameters) indicating a Pareto-frontier between the two. Previously, only the full-batch estimators (black markers) were known, of which only KFAC is both tractable and performant. Our derived estimators greatly increase the design space and provide many Pareto-optimal estimators, especially for lower runtime budgets.

Partitioning by class label. In Sec. 4.3, we hypothesized that grouping input-output sets by label information could improve bounds due to higher intra-class correlation than inter-class anti-correlation. Across all runs, this did not make a significant difference for learning invariances. However, for optimizing solely the prior precision (cf. App. C.1) it improves the bound and test log likelihood by 10% on average. This suggests that improved partitioning can help.

5.2. Benchmark of Proposed Estimators

We compare the proposed marginal likelihood estimators to the KFAC-Laplace, the state-of-the-art among Laplace approximations for hyperparameter optimization (Immer et al., 2021a; Daxberger et al., 2021; Antorán et al., 2022a). In the first setting, we optimize the layer-wise prior precision on a Wide ResNet 16-4 (Zagoruyko & Komodakis, 2016) on CIFAR-10 (Krizhevsky, 2009) and CIFAR-100 (Krizhevsky et al.). In the second setting, we additionally optimize learnable invariances, similar to data augmentation, where the Laplace-based method by Immer et al. (2022b) is extremely costly since it requires more than one pass through the dataset per gradient. The results in Table 1 suggest that stochastic marginal likelihood estimators are particularly useful for learning invariances, where they accelerate the runtime up to 25-fold at similar performance. In addition to our stochastic estimators based on subsets of data, i.e., NTK-150-1 and KFAC-150-1 with a batch size of 150 on single outputs, we show a full-dataset stochastic KFAC estimator that obtains a bound by sampling single random outputs (cf. remark in App. A).

5.3. Behavior with Varying Dataset Size

Since larger datasets theoretically lead to looser bounds when using constant subsets of data, we investigate how this impacts performance in Figure 7. In particular, we follow Immer et al. (2022b) and learn invariances present in transformed versions of CIFAR-10 (Krizhevsky, 2009) on random subsets ranging from 1 000 to all 50 000 data points. We follow prior work by considering distribution over affine invariances (Benton et al., 2020; van der Ouderaa & van der Wilk, 2021) detailed in App. C.4. We evaluate the test log likelihood of KFAC and NTK-based estimators on subsets of 400 data points and single independent outputs, KFAC-400-1 and NTK-400-1. We compare the approach with the method by Immer et al. (2022b), KFAC-full (equal to KFAC-N-C).

The cheaper marginal likelihood estimates and gradients allow us to do more hyperparameter updates per epoch given equivalent computational constraints. Compared to KFAC-full, we can therefore increase the number of hyperparameter steps for KFAC-400-1 and NTK-400-1 from 10 to 100, while still reducing overall training time. Figure 7 shows that our estimators perform on par or better than the full KFAC variant in terms of test log likelihood, especially, but not only, for small dataset sizes. We hypothesize for invariance learning that the ability to do more gradient updates is more beneficial than having tighter lower bounds. Further details on invariance learning, experimental details, and results can be found in App. C.4.

5.4. Scaling to Larger Datasets and Models

For invariance learning, the results on CIFAR-100 in Table 1 already indicated that KFAC-N-C is not scalable enough to enable gradient-based hyperparameter optimization. Mlodozeniec et al. (2023) also found that it is not possible to run it on the even larger TinyImagenet dataset (Le & Yang, 2015), which has $N = 100\,000$ data points and $C = 200$ classes, using a ResNet-50 with roughly 23 million parameters. Using the two fastest estimators for invariance learning, we show in Table 2 that our subset-of-data estimators enable optimizing hyperparameters in this setting.

h	Estimator	log likelihood	accuracy [%]
prior	BASELINE	-4.42 ± 0.08	45.6 ± 0.3
	NTK-400-1	-4.16 ± 0.25	49.8 ± 0.2
	KFAC-400-1	**-2.17 ± 0.02**	**53.0 ± 0.1**
invariance	AUGERINO	-	41.1 ± 0.2
	PARTITIONED	-	48.6 ± 0.0
	NTK-70-1	-3.15 ± 0.08	56.3 ± 0.1
	KFAC-60-1	**-2.02 ± 0.10**	**58.4 ± 0.3**

Table 2. Hyperparameter learning on TinyImagenet with ResNet-50 in comparison to Augerino (Benton et al., 2020) and neural network partitioning taken from Mlodozeniec et al. (2023). While previous Laplace approximations are intractable in this setting, KFAC with a subset size of 60 data points on single outputs excels.

We compare to the results provided by Mlodozeniec et al. (2023) and use the same architecture but with Fixup (Zhang et al., 2019b) instead of normalization layers, which would be incompatible with weight decay (Antorán et al., 2022b). Optimizing layer-wise prior precisions can greatly improve over the baseline with default settings and no data augmentation. Further, our estimators improve over invariance learning using neural network partitioning (Mlodozeniec et al., 2023) and Augerino (Benton et al., 2020).

5.5. Practical Considerations

In our experiments, the proposed estimators using subset-of-data lower bounds using a single output often perform best, in particular using the parametric KFAC variant. Although, the NTK-based variant often yields a tighter bound, KFAC-based bounds seem particularly well-suited for marginal-likelihood optimization and are therefore preferable in practice. As illustrated through invariance learning, our stochastic estimators are efficient for general differentiable hyperparameter optimization, which is an exciting future direction.

6. Related Work

Marginal-likelihood optimization, also referred to as *empirical Bayes*, is the *de-facto* standard for hyperparameter optimization of Gaussian process models (Rasmussen & Williams, 2006) without validation data, and was used in the early days of Bayesian neural networks (MacKay, 1992b; Foresee & Hagan, 1997). Use in modern larger neural networks dwindled, with Blundell et al. (2015) reporting failure using mean-field variational approximations, although the approach continued to work in Deep Gaussian Processes (Damianou & Lawrence, 2013; Dutordoir et al., 2020), and newer work demonstrates the feasibility in modern neural networks (Ober & Aitchison, 2021; Immer et al., 2021a). Antorán et al. (2022c) recently proposed an interesting alternative based on sampling, which works for prior precisions but not general hyperparameters. In the context of deep learning, the marginal likelihood has so far been used to select regularization strength, invariances, architectures, and representations. Our derived estimators further extend the family of Laplace approximations and provide alternatives to their parametric approximations. The partitioning of the kernel in our estimators is similar to the Bayesian committee machine (Tresp, 2000; Deisenroth & Ng, 2015), which distributes Gaussian process inference into smaller kernels but does not give a valid lower bound on the marginal likelihood. For a detailed discussion on the benefits, but also issues, of the marginal likelihood, we refer to Gelman et al. (1995, Sec. 7).

NTK and GGN Jacot et al. (2018) introduced the neural tangent kernel to characterize training dynamics of neural networks under squared loss. The training dynamics are even available in a closed-form when considering infinite width. However, computing kernels in this case is not trivial for common architectures (Novak et al., 2019). Our estimators use the NTK of neural networks at finite width, sometimes referred to as empirical NTK (Novak et al., 2022). Interestingly, the NTK and Gauss-Newton approximation to the Hessian are dual to each other as shown by (Khan et al., 2019), and Immer et al. (2021b) for general likelihoods. Our work builds on this duality and derives novel estimators from the NTK viewpoint that allow stochastic estimation. The NTK is also used to estimate generalization without the marginal likelihood, e.g., in the context of neural architecture search (Park et al., 2020; Chen et al., 2021). Further, the linearized Laplace with NTK been used to improve posterior predictives (Deng et al., 2022; Kim et al., 2023).

7. Conclusion

In this paper, we have derived stochastic estimators for the linearized Laplace approximation to the marginal likelihood that are suitable objectives for stochastic-gradient based optimization of hyperparameters. Our estimators are derived from a functional view of the Laplace using the neural tangent kernel and allow to trade off estimation accuracy and speed. Our experiments show that the (mini-)batch estimators perform on par with previous full-batch estimators but are many times faster. This suggests that they could be useful for learning more complex hyperparameters with SGD. Future research could further find ways to make bounds tighter, for example, by improving partitioning of the NTK. Further, it could be interesting to apply the fast estimators to large-scale Bayesian linear models.

Acknowledgements

A.I. gratefully acknowledges funding by the Max Planck ETH Center for Learning Systems (CLS). The authors thank the reviewers for their constructive feedback and comments, in particular R2, who suggested Figures 4, 5, and 6.

References

Antorán, J., Barbano, R., Leuschner, J., Hernández-Lobato, J. M., and Jin, B. A probabilistic deep image prior for computational tomography. *arXiv preprint arXiv:2203.00479*, 2022a.

Antorán, J., Janz, D., Allingham, J. U., Daxberger, E., Barbano, R. R., Nalisnick, E., and Hernández-Lobato, J. M. Adapting the linearised laplace model evidence for modern deep learning. In *International Conference on Machine Learning*, pp. 796–821. PMLR, 2022b.

Antorán, J., Padhy, S., Barbano, R., Nalisnick, E., Janz, D., and Hernández-Lobato, J. M. Sampling-based inference for large linear models, with application to linearised laplace. *arXiv preprint arXiv:2210.04994*, 2022c.

Benton, G., Finzi, M., Izmailov, P., and Wilson, A. G. Learning invariances in neural networks. *arXiv preprint arXiv:2010.11882*, 2020.

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

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

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. and Martens, J. KFAC-JAX, 2022. URL http://github.com/deepmind/kfac-jax.

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.

Chen, W., Gong, X., and Wang, Z. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *arXiv preprint arXiv:2102.11535*, 2021.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999. PMLR, 2016.

Damianou, A. and Lawrence, N. D. Deep Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, volume 31, pp. 207–215, Scottsdale, Arizona, USA, 29 Apr–01 May 2013. PMLR.

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

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., and Hennig, P. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Deisenroth, M. and Ng, J. W. Distributed gaussian processes. In *International Conference on Machine Learning*, pp. 1481–1490. PMLR, 2015.

Deng, Z., Zhou, F., and Zhu, J. Accelerated linearized laplace approximation for bayesian deep learning. In *Advances in Neural Information Processing Systems*, 2022.

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 (AISTATS)*, Aug 2020.

Fischer, E. Über den hadamardschen determinantensatz. *Arch. Math. U. Phys. (3)*, 13:32–40, 1908.

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.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. *Bayesian data analysis*. Chapman and Hall/CRC, 1995.

Hataya, R. and Nakayama, H. Gradient-based hyperparameter optimization without validation data for learning fom limited labels. 2021.

Horn, R. A. and Johnson, C. R. *Matrix analysis*. Cambridge university press, 2012.

Hutter, F., Kotthoff, L., and Vanschoren, J. (eds.). *Automated Machine Learning - Methods, Systems, Challenges*. Springer, 2019.

Immer, A., Bauer, M., Fortuin, V., Rätsch, G., and Emtiyaz, K. M. Scalable marginal likelihood estimation for model selection in deep learning. In *International Conference on Machine Learning*, pp. 4563–4573. PMLR, 2021a.

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, 2021b.

Immer, A., Torroba Hennigen, L., Fortuin, V., and Cotterell, R. Probing as quantifying inductive bias. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 1839–1851, 2022a.

Immer, A., van der Ouderaa, T. F., Ratsch, G., Fortuin, V., and van der Wilk, M. Invariance learning in deep neural networks with differentiable laplace approximations.

In *Advances in Neural Information Processing Systems*, 2022b.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pp. 8571–8580, 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.

Kim, S., Park, S., Kim, K., and Yang, E. Scale-invariant bayesian neural networks with connectivity tangent kernel. 2023.

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

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-100 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010. URL http://yann.lecun.com/exdb/mnist/.

Lorraine, J., Vicol, P., and Duvenaud, D. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pp. 1540–1552. PMLR, 2020.

MacKay, D. J. Bayesian model comparison and backprop nets. In *Advances in neural information processing systems*, pp. 839–846, 1992a.

MacKay, D. J. The evidence framework applied to classification networks. *Neural computation*, 4(5):720–736, 1992b.

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.

Martens, J. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

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

Mlodozeniec, B., Reisser, M., and Louizos, C. Hyperparameter optimization through neural network partitioning. *arXiv preprint arXiv:2304.14766*, 2023.

Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Novak, R., Xiao, L., Hron, J., Lee, J., Alemi, A. A., Sohl-Dickstein, J., and Schoenholz, S. S. Neural tangents: Fast and easy infinite neural networks in python. *arXiv preprint arXiv:1912.02803*, 2019.

Novak, R., Sohl-Dickstein, J., and Schoenholz, S. S. Fast finite width neural tangent kernel. In *International Conference on Machine Learning*, pp. 17018–17044. PMLR, 2022.

Ober, S. W. and Aitchison, L. Global inducing point variational posteriors for bayesian neural networks and deep gaussian processes. In *International Conference on Machine Learning*, pp. 8248–8259. PMLR, 2021.

Ober, S. W., Rasmussen, C. E., and van der Wilk, M. The promises and pitfalls of deep kernel learning. In *Uncertainty in Artificial Intelligence (UAI)*, volume 161, pp. 1206–1216. PMLR, 2021.

Osawa, K. Automatic Second-Order Differentiation Library (ASDL), 2021. URL http://github.com/kazukiosawa/asdl.

Osawa, K., Tsuji, Y., Ueno, Y., Naruse, A., Yokota, R., and Matsuoka, S. Large-scale distributed second-order optimization using kronecker-factored approximate curvature for deep convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12359–12367, 2019.

Park, D. S., Lee, J., Peng, D., Cao, Y., and Sohl-Dickstein, J. Towards nngp-guided neural architecture search. *arXiv preprint arXiv:2011.06006*, 2020.

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.

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

Schwöbel, P., Jørgensen, M., Ober, S. W., and van der Wilk, M. Last layer marginal likelihood for invariance learning. In *Proceedings of the Twenty Fifth International Conference on Artificial Intelligence and Statistics*, 2022.

Titsias, M. Variational learning of inducing variables in sparse gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, volume 5, pp. 567–574. PMLR, 16–18 Apr 2009.

Tresp, V. A bayesian committee machine. *Neural computation*, 12(11):2719–2741, 2000.

van der Ouderaa, T. F. and van der Wilk, M. Learning invariant weights in neural networks. In *Workshop in Uncertainty & Robustness in Deep Learning, ICML*, 2021.

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.

Wang, R., Walters, R., and Yu, R. Approximately equivariant networks for imperfectly symmetric dynamics. In *International Conference on Machine Learning*, pp. 23078–23091. PMLR, 2022.

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

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.

A. Theoretical Results and Proofs

The following Theorem and corollary are required to bound the parametric and NTK-based marginal likelihood approximations as they allow to bound the determinant, or the log-determinant, respectively by ignoring off-diagonal blocks of a matrix.

Theorem 4 (Fischer's inequality (1908))**.** *For a positive definite matrix* $\mathbf{M}$*, as defined hereafter, it holds that*

$$\det \mathbf{M} = \det \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{bmatrix} \leq \det \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{C} \end{bmatrix} = \det(\mathbf{A} \oplus \mathbf{C}) = \det \mathbf{A} \det \mathbf{C}. \tag{15}$$

This Theorem is from Fischer (1908). We refer to Horn & Johnson (2012) for a proof. We immediately have the following useful corollary negative log-determinants, which show up in the Laplace approximation to the log marginal likelihood. The result simply follows from the fact that the $\log$ is monotonically increasing.

Corollary 4.1. *For a positive definite matrix* $\mathbf{M}$ *as defined in Theorem 4, the following inequality holds*

$$-\log\det \mathbf{M} = -\log\det \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \geq -\log\det \mathbf{A} - \log\det \mathbf{C}. \tag{16}$$

For our bounds, a more general partitioned form of $\mathbf{M}$ is required, which corresponds to repeated bounding with Theorem 4 through partitions. In particular, we deal with a matrix $\mathbf{M} \in \mathbb{R}^{M \times M}$ and use indices 1 through M, i.e., $[M] = \{1, \dots, M\}$ that, for example, allow to denote the mth diagonal element as $\mathrm{M}_{m,m}$. Further, we introduced a notation based on index sets in Sec. 2 that allows to denote blocks or off-diagonal elements of $\mathbf{M}$. In particular, let $\mathcal{A} \subset [M]$ and $\mathcal{A}' \subset [M]$ disjoint subsets, $\mathcal{A} \cap \mathcal{A}' = \emptyset$ of the indices denoting dimensions in $\mathbf{M}$, then we write $\mathbf{M}_{\mathcal{A},\mathcal{A}} = \mathbf{M}_{\mathcal{A}} \in \mathbb{R}^{|\mathcal{A}| \times |\mathcal{A}|}$ for the square block matrix with entries $\mathrm{M}_{i,j}$ such that $i \in \mathcal{A}$ and $j \in \mathcal{A}$. In line with this, off-diagonal blocks are $\mathbf{M}_{\mathcal{A},\mathcal{A}'} \in \mathbb{R}^{|\mathcal{A}| \times |\mathcal{A}'|}$.

Partitioning the dimension indices $[M]$ of $\mathbf{M}$ into two disjoint subsets $\mathcal{A}$ and $\mathcal{A}'$, we have three blocks, $\mathbf{M}_{\mathcal{A}}, \mathbf{M}_{\mathcal{A}'}$, and $\mathbf{M}_{\mathcal{A},\mathcal{A}'}$. Since we can re-order the matrix to conform to such blocks, because it relies on simultaneous permutation of rows and columns, we can apply Theorem 4 to $\mathbf{M}$ and have

$$\det \mathbf{M} = \det \begin{bmatrix} \mathbf{M}_{\mathcal{A}} & \mathbf{M}_{\mathcal{A},\mathcal{A}'} \\ \mathbf{M}_{\mathcal{A}',\mathcal{A}} & \mathbf{M}_{\mathcal{A}'} \end{bmatrix} \leq \det \begin{bmatrix} \mathbf{M}_{\mathcal{A}} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{\mathcal{A}'} \end{bmatrix} = \det(\mathbf{M}_{\mathcal{A}} \oplus \mathbf{M}_{\mathcal{A}'}) = \det \mathbf{M}_{\mathcal{A}} + \det \mathbf{M}_{\mathcal{A}'}. \tag{17}$$

Note that the first equality here only holds for the determinant and not for its matrix argument since the partitioning by indices permutes the rows and columns simultaneously, i.e., an even amount of times. With this, we have the following Lemma

Lemma 1. *For any partitioning* $\mathcal{P} = \{\mathcal{P}_s\}_{s=1}^S$ *of dimension indices* $[M]$ *of the positive definite matrix* $\mathbf{M}$*, we have*

$$\det \mathbf{M} = \det \begin{bmatrix} \mathbf{M}_{\mathcal{P}_1} & \mathbf{M}_{\mathcal{P}_1,\mathcal{P}_2} & \cdots & \mathbf{M}_{\mathcal{P}_1,\mathcal{P}_S} \\ \mathbf{M}_{\mathcal{P}_2,\mathcal{P}_1} & \mathbf{M}_{\mathcal{P}_2,\mathcal{P}_2} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{M}_{\mathcal{P}_S,\mathcal{P}_1} & \cdots & \cdots & \mathbf{M}_{\mathcal{P}_S} \end{bmatrix} \leq \det(\mathbf{M}_{\mathcal{P}_1} \oplus \cdots \oplus \mathbf{M}_{\mathcal{P}_S}) = \prod_{s=1}^{S} \det \mathbf{M}_{\mathcal{P}_s}, \tag{18}$$

and further refining *the partitioning* $\mathcal{P}$ *into* $\mathcal{P}'$*, such that each element in* $\mathcal{P}'$ *is a subset of* $\mathcal{P}'$ *is a futher upper bound.*

Proof. To obtain this bound, we iteratively apply Theorem 4. In particular, let $\mathcal{R}_s$ be the complement of the first s subsets, i.e., $\mathcal{R}_s = \{\mathcal{P}_1, \dots, \mathcal{P}_s\}^{\complement}$, for example, $\mathcal{R}_1 = \mathcal{P}_1^{\complement}$ and $\mathcal{R}_S = \emptyset$.[6] The complement $(\cdot)^{\complement}$ here is with respect to the full set of indices $[M]$. Using Theorem 4 S times on these complementing sets, we have

$$\begin{aligned} \det \mathbf{M} = \det \begin{bmatrix} \mathbf{M}_{\mathcal{P}_1} & \mathbf{M}_{\mathcal{P}_1,\mathcal{R}_1} \\ \mathbf{M}_{\mathcal{R}_1,\mathcal{P}_1} & \mathbf{M}_{\mathcal{R}_1} \end{bmatrix} &\leq \det \mathbf{M}_{\mathcal{P}_1} \det \mathbf{M}_{\mathcal{R}_1} = \det \mathbf{M}_{\mathcal{P}_1} \det \begin{bmatrix} \mathbf{M}_{\mathcal{P}_2} & \mathbf{M}_{\mathcal{P}_2,\mathcal{R}_2} \\ \mathbf{M}_{\mathcal{R}_2,\mathcal{P}_2} & \mathbf{M}_{\mathcal{R}_2} \end{bmatrix} \\ &\leq \det \mathbf{M}_{\mathcal{P}_1} \det \mathbf{M}_{\mathcal{P}_2} \det \mathbf{M}_{\mathcal{R}_2} = \cdots \leq \prod_{s=1}^{S} \det \mathbf{M}_{\mathcal{P}_s}. \end{aligned}$$

This proves the main statement of the Lemma. By refining the partition, we simply extend the upper bounds since a refinement splits up each element in $\mathcal{P}$ into at least one subset, which again enables application of Theorem 4. □

[6] The order $1, \dots, S$ is arbitrary.

Theorem 1 (Parametric structured lower bound)**.** *For any partitioning $\mathcal{P}$ of the parameter indices $\{1, .., P\}$, the block-diagonal approximation of $\mathbf{H}_\mathcal{P}$ to $\mathbf{H}$ results in a lower bound to the* LA *log marginal likelihood (Equation 5), i.e.,*

$$\log q_{\tilde{\mathbf{w}}}(\mathcal{D}|\mathbf{h}) \geq \log p(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) - \tfrac{1}{2}\log|\mathbf{H}_\mathcal{P} + \mathbf{P}_0| + c, \tag{9}$$

where $c = \frac{P}{2}\log 2\pi$. Further, a refinement[7] of the partition $\mathcal{P}$ to $\mathcal{P}'$ will result in a lower bound with even more slack.

Proof. Since the log-likelihood and log-prior terms are identical, we only need to inspect the relationship between $-\log|\mathbf{H}+\mathbf{P}_0|$ and $-\log|\mathbf{H}_\mathcal{P}+\mathbf{P}_0|$. Due to the assumption that $\mathbf{P}_0$ is diagonal, it can be added to the individual blocks of $\mathbf{H}_\mathcal{P}$ since a diagonal is the most *refined* partition possible on the indices $\{1, \dots, P\}$. [8] The lower bound on the negative log determinant and the increased slack when refining then follows from Lemma 1, which upper bounds the product of determinants, and thus lower bounds the negative log determinant (Corollary 4.1). □

Theorem 2 (Data subset lower bound)**.** *For any partitioning $\mathcal{B}$ of inputs and outputs $\mathcal{I}$, the corresponding block-diagonal* NTK *approximation $\mathbf{K}_\mathcal{B}$ to $\mathbf{K}$ results in a lower bound to the linearized Laplace marginal likelihood, i.e.,*

$$\begin{aligned}\log q_{\tilde{\mathbf{w}}}(\mathcal{D}|\mathbf{h}) &\geq \log p(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) - \tfrac{1}{2}\log|\mathbf{K}_\mathcal{B} + \mathbf{I}||\mathbf{P}_0| + c \\ &\propto \log p(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) - \tfrac{1}{2}\log|\mathbf{P}_0| - \tfrac{1}{2}\textstyle\sum_{m=1}^{M}\log|\mathbf{K}_{\mathcal{B}_m} + \mathbf{I}|,\end{aligned} \tag{11}$$

where $c = \frac{P}{2}\log 2\pi$ is left out in the second line indicated by the proportionality. Further, a refinement of partition $\mathcal{B}$ to $\mathcal{B}'$ will result in a lower bound with more slack.

Proof. As shown by Immer et al. (2021a) and in Sec. 2, the linearized Laplace approximation can be written in NTK-form using the matrix determinant Lemma giving

$$\begin{aligned}\log q_{\tilde{\mathbf{w}}}(\mathcal{D}|\mathbf{h}) &= \log p(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) - \tfrac{1}{2}\log|\mathbf{J}_{\tilde{\mathbf{w}}}\mathbf{P}_0^{-1}\mathbf{J}_{\tilde{\mathbf{w}}}^\intercal\mathbf{\Lambda}_{\tilde{\mathbf{w}}} + \mathbf{I}||\mathbf{P}_0| + \tfrac{P}{2}\log 2\pi \\ &= \log q(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) - \tfrac{1}{2}\log|\mathbf{P}_0| + \tfrac{P}{2}\log 2\pi - \tfrac{1}{2}\log|\mathbf{K} + \mathbf{I}| \\ &\geq \log q(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) - \tfrac{1}{2}\log|\mathbf{P}_0| + \tfrac{P}{2}\log 2\pi - \tfrac{1}{2}\log|\mathbf{K}_\mathcal{B} + \mathbf{I}| \\ &= \log q(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) - \tfrac{1}{2}\log|\mathbf{P}_0| + \tfrac{P}{2}\log 2\pi - \tfrac{1}{2}\textstyle\sum_{m=1}^{M}\log|\mathbf{K}_{\mathcal{B}_m} + \mathbf{I}|,\end{aligned}$$

where we first re-order and use our definition of the scaled NTK (cf. Sec. 2). Then, we use the lower bound according to Lemma 1 further giving us the statement that refinement leads to more slack. We again note that adding a diagonal, in this case $\mathbf{I}$ to $\mathbf{K}$, can simply be absorbed into the block-matrices and extracted afterwards when applying it. In particular, define $\hat{\mathbf{K}} = \mathbf{K} + \mathbf{I}$, apply the bound, and we obtain $\hat{\mathbf{K}}_{\mathcal{B}_m} = \mathbf{K}_{\mathcal{B}_m} + \mathbf{I}$ for all m and thus $\hat{\mathbf{K}}_\mathcal{B} = \mathbf{K}_\mathcal{B} + \mathbf{I}$. □

Theorem 3 (Parametric data subset bound)**.** *The* NTK*-based lower bound in Equation 11 to the linearized Laplace marginal likelihood is equivalent to a parametric variant,*

$$\begin{aligned}&\log p(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) - \tfrac{1}{2}\log|\mathbf{P}_0| - \tfrac{1}{2}\textstyle\sum_{m=1}^{M}\log|\mathbf{K}_{\mathcal{B}_m} + \mathbf{I}| \\ &= \log p(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) + \tfrac{M\text{-}1}{2}\log|\mathbf{P}_0| \text{-} \tfrac{1}{2}\textstyle\sum_{m=1}^{M}\log|\mathbf{H}_{\mathcal{B}_m} + \mathbf{P}_0|,\end{aligned} \tag{13}$$

where $\mathbf{H}_{\mathcal{B}_m} = \mathbf{J}_{\mathcal{B}_m}^\intercal\mathbf{\Lambda}_{\mathcal{B}_m}\mathbf{J}_{\mathcal{B}_m}$ is estimated on a data subset.

Proof. The idea is to apply the matrix determinant Lemma to move from the subset of data NTK bound back to a parametric variant. This gives us a log determinant that depends on the GGN defined on subsets of input-output pairs. We subtract the log joint, $\log p(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h})$, and add $\frac{1}{2}\log|\mathbf{P}_0|$ from to both sides of Equation 13 and multiply by 2 to abbreviate the following equations. We then have

$$\begin{aligned}-\textstyle\sum_{m=1}^{M}\log|\mathbf{K}_{\mathcal{B}_m} + \mathbf{I}| &= -\textstyle\sum_{m=1}^{M}\log|\mathbf{J}_{\mathcal{B}_m}\mathbf{P}_0^{-1}\mathbf{J}_{\mathcal{B}_m}^\intercal\mathbf{\Lambda}_{\mathcal{B}_m} + \mathbf{I}| \\ &= M\log|\mathbf{P}_0| - \textstyle\sum_{m=1}^{M}\log|\mathbf{J}_{\mathcal{B}_m}\mathbf{P}_0^{-1}\mathbf{J}_{\mathcal{B}_m}^\intercal\mathbf{\Lambda}_{\mathcal{B}_m} + \mathbf{I}||\mathbf{P}_0| \\ &= M\log|\mathbf{P}_0| - \textstyle\sum_{m=1}^{M}\log|\mathbf{J}_{\mathcal{B}_m}^\intercal\mathbf{\Lambda}_{\mathcal{B}_m}\mathbf{J}_{\mathcal{B}_m} + \mathbf{P}_0| \\ &= M\log|\mathbf{P}_0| - \textstyle\sum_{m=1}^{M}\log|\mathbf{H}_{\mathcal{B}_m} + \mathbf{P}_0|,\end{aligned}$$

[7] A refinement $\mathcal{P}'$ of $\mathcal{P}$ is a partition such that for all $\mathcal{P}'_s \in \mathcal{P}'$ there exists $\mathcal{P}_s \in \mathcal{P}$ such that $\mathcal{P}'_s \subseteq \mathcal{P}_s$.

[8] In case, $\mathbf{P}_0$ would not be diagonal and its structure would not correspond to a refinement of $\mathcal{P}$, one would have to treat it jointly with $\mathbf{H}$ and apply the bound to $\hat{\mathbf{H}} \stackrel{\text{def}}{=} \mathbf{H} + \mathbf{P}_0$ and then $\hat{\mathbf{H}}_\mathcal{P}$.

where $\mathbf{J}_{\mathcal{B}_m}$ corresponds to the Jacobians only for the input-output pairs in $\mathcal{B}_m$ and $\mathbf{\Lambda}_{\mathcal{B}_m}$ to the log-likelihood Hessian for these outputs (cf. Sec. 2.3). subtracting and adding the removed quantities again concludes the proof. □

Remark. The form of Equation 13, in particular $\mathbf{H}_{\mathcal{B}_m}$ is reminiscent of a GGN approximation but just on a subset of input-output pairs. In particular, it depends on the subset structure chosen for the NTK bound, and could be a class-wise GGN or simply on a random subset of data. Subsets of data do not change the shape of the parametric estimator, i.e., it still requires the log determinant of a $P \times P$ matrix but computing that matrix can be greatly sped up: apart from the requirement to compute Jacobians for all N data points, a major issue with the GGN approximation is that its cost scales linearly with the outputs C as well (Botev et al., 2017). Therefore, it is often approximated using sampling via equality to the Fisher information (Martens & Grosse, 2015) for exponential family likelihoods of natural form (Martens, 2020). By partitioning the input-output pairs $[NC]$ into C partitions, we can compute the GGN for a single output and use it as a proper lower-bound to the LA marginal likelihood at the same cost as the Fisher, which does not lead a lower bound of the LA due to the required sampling approximation. Further, we can apply our parametric bound in Theorem 1 to use mixed bounds on subsets of data and subsets of parameter groups enabling, for example, a lower bound to the LA marginal likelihood using KFAC on a subset of data.

Corollary 3.1 (Parametric doubly lower bound)**.** *For any partitioning $\mathcal{B}$ of input-output pairs $\mathcal{I}$ and $\mathcal{P}$ of parameter indices $\{1, .., P\}$, respectively, the following lower bound to the linearized Laplace marginal likelihood holds:*

$$\begin{aligned}\log q_{\tilde{\mathbf{w}}}(\mathcal{D}|\mathbf{h}) \geq \log p(\mathcal{D}, \tilde{\mathbf{w}}|\mathbf{h}) + \tfrac{M-1}{2} \log |\mathbf{P}_0| \\ - \tfrac{1}{2} \textstyle\sum_{m=1}^{M} \log |\mathbf{H}_{\mathcal{B}_m,\mathcal{P}} + \mathbf{P}_0| + \tfrac{P}{2} \log 2\pi,\end{aligned} \tag{14}$$

where $\mathbf{H}_{\mathcal{B}_m,\mathcal{P}}$ is a block-structured GGN *approximation on subsets of input-output pairs.*

Proof. This corollary simply follows from applying the parametric lower bound, Theorem 1, to the parametric estimator on a subset of data in Theorem 3. □

B. Computational Considerations and Complexity

The computational complexity of the proposed estimators as well as the GGN, NTK, and their corresponding approximations like KFAC greatly depend on the model architecture. For simplicity, we assume a neural network with P parameters in L linear hidden layers with each width of D, inputs $\mathbf{x} \in \mathbb{R}^D$ of the same dimensionality, with output dimensionality C, i.e., we have $P = DC + \sum_{l \in [L]} D^2$, where DC are the parameters of the linear output layer. Further, we have N data points. The complexity of automatically differentiating the log determinant of the GGN or NTK is equivalent to computing the log determinant of these matrices defined in Equation 7. Computing the log determinant of the GGN first requires computing the GGN itself, which is $\mathcal{O}(NP^2C)$ for computing the sum of N Jacobian outer products, where each Jacobian is $C \times P$. The log determinant is then additionally $\mathcal{O}(P^3)$. In comparison, for naive computation of the NTK log determinant, we compute the $NC \times NC$ kernel in $\mathcal{O}(N^2C^2P)$ and its determinant in $\mathcal{O}(N^3C^3)$. The complexity for KFAC-GGN is $\mathcal{O}(NLD^2C)$ for summing N outer products for L layers and its determinant $\mathcal{O}(D^3)$ for each layer giving $\mathcal{O}(LD^3)$. We have the following computational and complexities for computing and differentiating the log determinant using automatic differentiation:

$$\text{GGN} \in \mathcal{O}(NP^2C + P^3) \quad \text{NTK} \in \mathcal{O}(N^2C^2P + N^3C^3) \quad \text{KFAC-GGN} \in \mathcal{O}(NLD^2C + LD^3). \tag{19}$$

While these are for naive estimators that can be improved for certain architectures, e.g., see Novak et al. (2022) for faster NTK computations, these complexities describe the worst-case setting.

Our proposed lower bounds based on subsets of inputs and outputs can greatly improve these complexities. In particular, we can replace N by a subset size $M \ll N$ and use output-wise bounds to obtain $C = 1$. In our experiments, the fastest and still performant methods are NTK-M-1 and KFAC-M-1, which have a computational and memory complexity for estimation and automatic differentiation of $\mathcal{O}(M^2P + M^3)$ and $\mathcal{O}(MLD^2 + LD^3)$, respectively. Depending on the architecture, the NTK estimator can further be faster than $\mathcal{O}(M^2P)$. Overall, the proposed methods attain an acceleration at least linear in the number of outputs C and data points N.

C. Additional Experimental Results and Details

We provide additional experimental details and results complementing those in the main text here. For our implementation, we modify and extend the `asdl` library (Osawa, 2021) that offers fast computation of KFAC and NTK, as well as `laplace-torch` (Daxberger et al., 2021) for the marginal likelihood approximations. The code is available at https://github.com/AlexImmer/ntk-marglik.

C.1. Tightness of Bounds, Performance, and Runtime

The experiments illustrating the bounds in the main text described in Sec. 5.1 are conducted on randomly chosen 1000 MNIST subsets for three random seeds and further images are fully rotated at random, i.e., up to $\pm\pi$. We then use the LILA method proposed by Immer et al. (2022b) to try to learn these underlying invariances, which is essential to generalize well on this task since standard convolutional and fully connected layers are not rotationally invariant. We use a simple convolutional network with three layers, max pooling, and a linear classification head that totals roughly 16 000 parameters so that we can compute and differentiate the full GGN and NTK. We use 30 augmentation samples to average the outputs and optimize the network parameters, invariance parameters, and prior parameters with Adam (Kingma & Ba, 2015). For network parameters we use a learning rate of 10^{-3} and decay it to 10^{-9} using cosine decay and use a batch size of 250. The invariance and prior learning hyperparameters follow the settings of Immer et al. (2022b): 10 epochs burnin and then update hyperparameters every epoch with learning rates 0.1 and 0.05 for prior precision and invariance parameters, respectively. Both are decayed by a factor of 10 using cosine decay. We use an optimized network to assess the bounds in Figures 4 to 6. After convergence, we apply the different bounds and assess them over the entire grid of rotational invariance $\eta \in [0, \pi]$. Here, we additionally present results for hyperparameter optimization using the same bounds in Figure 8 and Figure 9. The experiments are run on an internal compute cluster with different NVIDIA GPUs. The timing experiments are run on a single A100 sequentially to ensure comparability.

In addition to the invariance learning experiments, we compare the bounds for varying values of a scalar prior precision in all illustrative figures in the main text. Here, we show results optimizing the layerwise prior precision in Figure 10, 11, and 13 again on 1000 randomly sampled MNIST digits per seed but without rotating them. When only learning prior precision parameters and not invariances, a slack in the bound translates more directly into a slack in the test performance as can be seen in the figures. However, it is worth noting that the NTK and KFAC estimators are significantly cheaper on large batches of data for just optimizing the prior precision, in comparison to learning invariances. In Figure 13, we see that stochastic NTK-based estimators are Pareto-optimal in many cases as it was the case for invariance learning. We also find that small batch sizes do not lead to problems at larger scale (cf. Sec. 5.2) where they perform on par with full dataset marginal likelihood estimators. In Figure 12, we show the effect of class-wise partitioning of batches on the bound in comparison to the random partition. We find that the bound can become tighter for both prior precision and invariance learning but incurs a slightly higher variance as indicated by the standard error displayed.

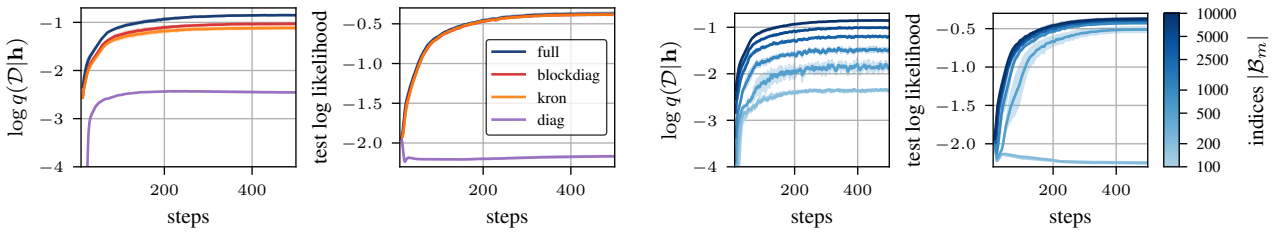

Figure 8. Parametric and NTK lower bound when optimizing invariances on random subsets of size 1000 from rotated MNIST (LeCun & Cortes, 2010). Only small subset sizes and the diagonal approximation fail to learn the invariance and result in good test log likelihood. This figure corresponds to Figure 3 and Figure 4 in the main text, which show the bounds constructed at step 500 with parameters and hyperparameters trained by the "full" variant.

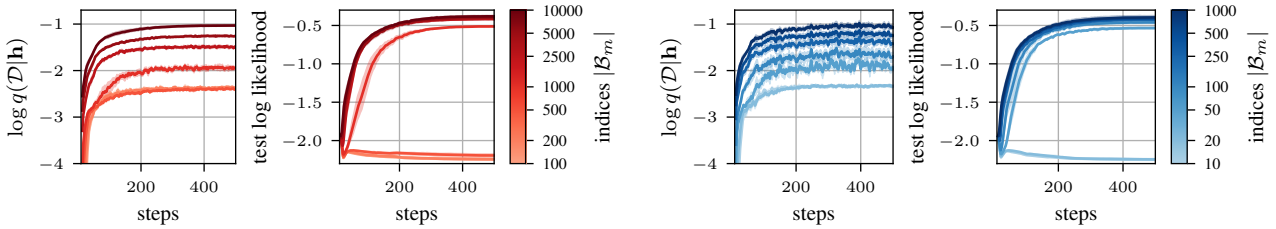

Figure 9. Doubly parametric subset-of-data bound on the left with block-diagonal approximation and class-wise NTK-based approximation. These figures correspond to Figure 5 and Figure 6 in the main text but use the bounds for optimization of invariances during training. In line with the bounds, already small subset sizes suffice to pick up the invariance in rotated MNIST and achieve good test log likelihood.

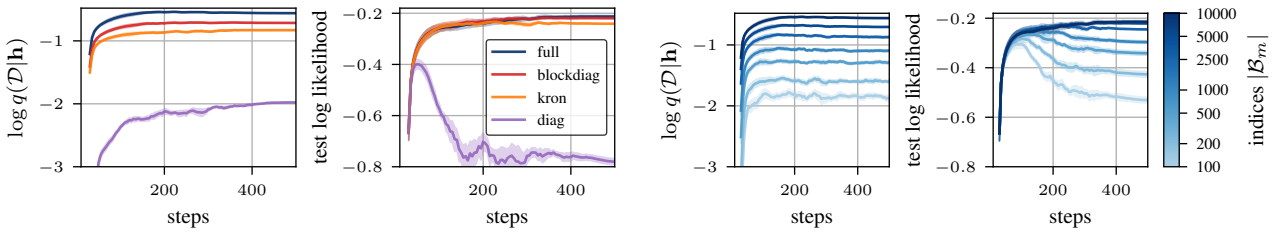

Figure 10. Parametric and NTK lower bound when optimizing prior precision and not invariances on random subsets of size 1000 from MNIST (LeCun & Cortes, 2010). In contrast to invariance learning, lower bounds result in more reduction in test performance for the NTK-based bounds that use stochastic estimates. This figure corresponds to Figure 3 and Figure 4 in the main text.

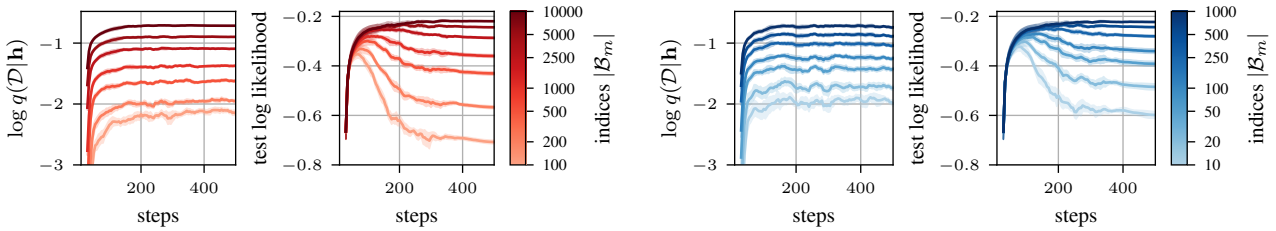

Figure 11. Doubly parametric subset-of-data bound with block-diagonal approximation (left) and class-wise NTK-based approximation (right) for prior precision optimization corresponding to Figure 5 and Figure 6 in the main text.

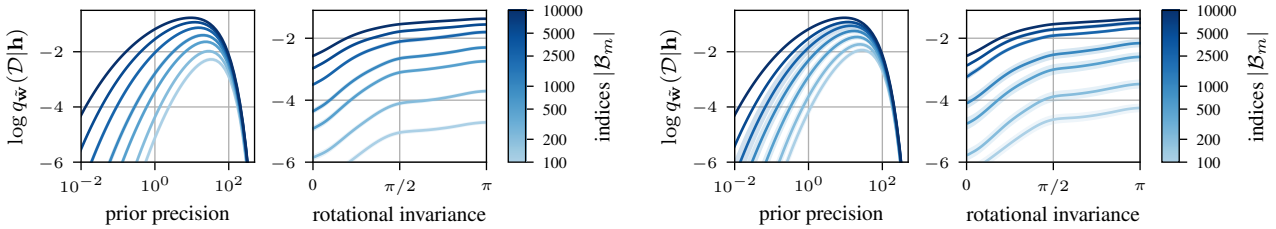

Figure 12. Effect of partitioning data by class labels (right) in comparison to random partitioning (left; same figure as Figure 4) as proposed in Sec. 4.3. Using the class labels can slighty redue the slack of the bound but tends to incur a larger variance in the estimated bound as indicated by the shaded regions denoting one standard error.

C.2. Additional Details for Estimator Benchmark on CIFAR

Marginal likelihood estimation with the linearized Laplace in previous works was commonly applied to ResNets on CIFAR-10 and CIFAR-100, where it gives remarkable performance improvements over unregularized networks in the case where we have no data augmentation or prior information. For example Immer et al. (2021a) and Daxberger et al. (2021) show that it improves the performance by 4% points over the baseline on CIFAR-10. Here, we reproduce this experiment and compare the performance of the previously used full-batch KFAC estimator, which produced the best results, to the proposed stochastic estimators. The results are given in Table 1 in the top block. Indeed, all estimators improve over the baseline that uses default settings of a Wide ResNet (Zagoruyko & Komodakis, 2016) but without data augmentation and Fixup initialization instead of normalization layers (Zhang et al., 2019b). We use the Wide ResNet 16-4 with Fixup (Zhang et al., 2019b), since a Gaussian prior conflicts with normalization layers (Zhang et al., 2019a; Antorán et al., 2022b). We optimize

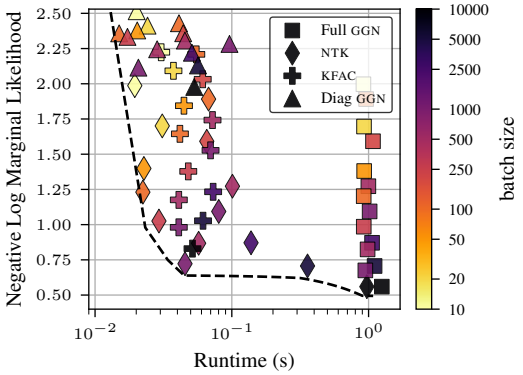

Figure 13. Pareto-frontier for prior precision learning runs on random subsets of 1000 MNIST digits with a small CNN. This figure corresponds to Figure 1 in the main text, which displays estimators for additional invariance learning, not only learning priors. Many of the NTK-based class-wise estimators are Pareto-optimal, i.e., the achieve the lowest marginal likelihood at a given runtime budget.

the network for 300 epochs with a batch size of 128 using SGD with momentum of 0.9 and optimize the prior precision every 5 epochs after a burnin phase of 10 epochs for with 50 gradient steps using a learning rate of 0.1 and Adam (Kingma & Ba, 2015) for both CIFAR-10 and CIFAR-100.

In the lower block of Table 1, we use our estimators in the context of learning invariance using laplace approximations (LILA, Immer et al., 2022b). We use the default hyperparameters of their largest-scale experiment, where they learn invariances with Wide ResNets using 20 augmentation samples. Their method uses full dataset KFAC for 200 epochs and takes more than 48 hours on of training on CIFAR-10 with an NVIDIA A100 GPU because they have to compute a preconditioner on the entire dataset for each hyperparameter gradient. On CIFAR-100, due to the scaling with output classes, it is therefore intractable as it would likely take weeks. The default settings from CIFAR-10 lead to out-of-memory errors. In contrast, our stochastic estimators can be directly evaluated on a small batch with a single output and thus improve the runtime by a factor 10 while maintaining the same performance, i.e., compare KFAC-150-1 with KFAC-N-C. Also our single-output KFAC bound already increases runtime by a factor of 5 and achieves the same performance. The choice of the batch size in our bounds is such that the GPU memory is utilized as much as possible. We therefore use NVIDIA A100 GPUs with 80GB memory to make the bounds as tight as possible and improve performance. To compare the runtimes, all methods use the same GPU type and runtimes are averaged over 3 or 5 seeds for invariance and prior learning, respectively.

C.3. Details on TinyImagenet Benchmark using ResNet-50

We largely follow the setup of Mlodozeniec et al. (2023) in this experiment and use their results on Augerino (Benton et al., 2020) and their own network partitioning method for hyperparameter optimization. The key difference is that we use Fixup (Zhang et al., 2019b) instead of normalization layers in the ResNet-50 architecture. Like Mlodozeniec et al. (2023), we use 16 channels in the first layer and the standard growth factor of 2 per layer. The network has roughly 23 000 000 parameters and therefore poses a complex task for differentiable hyperparameter optimization. Even with 80GB of GPU memory, we can only fit 60 to 70 data points for differentiable KFAC or NTK computations. However, the performance is still sufficient to improve over the baselines. We optimize the network weights for 100 epochs and, as in the case for CIFAR, decay the learning rate with a cosine schedule from 0.1 to 10^{-6}. The hyperparameters are optimized for 50 steps every epoch after 10 epochs of burn-in with a learning rate of 0.01 decayed to 0.001 with a cosine schedule. The learning rate for prior parameters is constant at 0.1. For only prior precision optimization, we use the default setting by Immer et al. (2021a) and update the prior parameters for 50 steps every 5 epochs after 10 epochs of burn-in.

C.4. Additional and Detailed Results for Invariance Learning on CIFAR-10 Subsets.

It is well known that soft or hard symmetry constraints are beneficial to machine learning methods and neural networks in particular, equivariant symmetries of (group-)convolutional layers being a canonical example (Cohen & Welling, 2016). Although explicitly embedding symmetry in architectures can be very effective, they need to be known and specified in advance and can not be adapted. The right symmetry can be difficult to choose and misspecified symmetries have the risk of restricting the capacity of a model to fit data (Wang et al., 2022). Symmetry discovery methods aim to automatically learn symmetries from available training data, which is a difficult task because symmetries provide constraints and are therefore not necessarily favorable in terms of typical training losses. In invariance learning, we parameterise a learnable distribution $\mathbf{g}(\mathbf{x};\boldsymbol{\eta})$ over possible transformations on inputs, defined by invariance parameters $\boldsymbol{\eta}$. By averaging outcomes of a regular non-invariant neural network $\mathbf{f}$ over inputs transformed by $\mathbf{g}$, we obtain a predictor $\tilde{f}$, which is (soft-) invariant to the transformations specified by $\boldsymbol{\eta}$. We might take S Monte Carlo samples $\boldsymbol{\epsilon}_1, \dots, \boldsymbol{\epsilon}_S \overset{\text{iid}}{\sim} p(\boldsymbol{\epsilon})$ (van der Wilk et al., 2018; Benton et al., 2020) to obtain an unbiased estimate of the invariant function $\hat{\mathbf{f}}$:

$$\hat{\mathbf{f}}(\mathbf{x};\mathbf{w},\boldsymbol{\eta}) = \mathbb{E}_{p(\mathbf{x}'|\mathbf{x},\boldsymbol{\eta})}[\mathbf{f}(\mathbf{x}';\mathbf{w})] = \mathbb{E}_{p(\boldsymbol{\epsilon})}[\mathbf{f}(\mathbf{g}(\mathbf{x},\boldsymbol{\epsilon};\boldsymbol{\eta});\mathbf{w})] \approx \tfrac{1}{S}\textstyle\sum_s \mathbf{f}(\mathbf{g}(\mathbf{x},\boldsymbol{\epsilon}_s;\boldsymbol{\eta});\mathbf{w})\,, \tag{20}$$

where $\hat{\mathbf{f}}$ is differentiable in both model parameters $\mathbf{w}$ as well as invariance parameters $\boldsymbol{\eta}$ through the reparameterisation trick (Kingma & Welling, 2013). In our experiments, we consider a combination of uniform distributions on affine generator matrices resulting in a 6-dimensional invariance parameter $\boldsymbol{\eta} \in \mathbb{R}^6$ that defines a density over the group of affine transformations, individually controlling x-translation, y-translation, rotation, x-scaling, y-scaling and shearing. Details on the used parameterisation can be found in Benton et al. (2020) and App. B of Immer et al. (2022b).

Jointly learning model parameters $\mathbf{w}$ and invariance parameters $\boldsymbol{\eta}$ is not a trivial exercise, as typical maximum likelihood solutions will always select the least restrictive no invariance in order to fit training data best, given a sufficiently flexible model. To overcome this issue, some people have considered the use of validation data, such as cross-validation or more sophisticated approaches, which may require expensive retraining or involved outer loops. Alternatively, Augerino (Benton et al., 2020) learns invariance solely on training data, but relies on explicit regularisation in order to do so, which requires knowledge of the used invariance parameterisation and additional tuning. For a discussion on particular failure cases of this approach we refer to Sec. 2. and App. C. of Immer et al. (2022b), which proposes to use scalable marginal likelihood estimates to learn invariance parameters. The marginal likelihood offers a principled way to learn invariances through Bayesian model selection, balancing data fit and model complexity through an Occam's razor effect. Unlike prior works that rely on the true marginal likelihood (van der Wilk et al., 2018) or variational lower bounds thereof (van der Wilk et al., 2018; van der Ouderaa & van der Wilk, 2021), modern differentiable Laplace approximations of the marginal likelihood can scale to large datasets and deep neural networks, such as ResNets. We consider this as a baseline, and denote it by KFAC-full as it uses the full dataset to estimate the marginal likelihood, which leads to a significant slowdown in training compared to classical MAP training. To improve estimates, we have derived lower bounds to the marginal likelihood that can be computed much more cheaply on subsets of the training data. This allows us to take more hyperparameter gradient steps per epoch at equivalent or faster training time. We hypothesize that taking more gradient steps can be beneficial, even though individual marginal likelihood estimates provide looser bounds.

We empirically evaluate performance of our method in conjunction with invariance learning on subsets of the same transformed versions of CIFAR-10 as used in Immer et al. (2022b). The use of reduced dataset sizes is a common data efficiency benchmark for invariance learning methods (Schwöbel et al., 2022). We measure performance for varying total dataset sizes where we estimate marginal likelihoods on fixed subsets of 400 datapoints and use independent outputs, KFAC-400-1 and NTK-400-1. We compare to KFAC-full which always uses the full set of available training data points for its estimates. We use a ResNet architecture with Fixup (Zhang et al., 2019b), to prevent Gaussian prior conflicts with normalization layers (Zhang et al., 2019a; Antorán et al., 2022b), and take S=20 Monte Carlo of invariance transformations. We optimize the network for 200 epochs with a batch size of 128 using Adam (Kingma & Ba, 2015) with cosine annealed learning rates starting at 0.1 for parameters and prior variances and invariance parameter learning rates starting at 0.05 for KFAC-full and 0.01 for KFAC-400-1 and KFAC-400-1, which use more gradient steps.

In the figures below, we show the marginal likelihood estimate (Figure 14), test log-likelihood (Figure 15) and test accuracy (Figure 16). We find that marginal likelihood estimates of KFAC-full are higher, which is not surprising since KFAC-400-1 and KFAC-400-1 use looser bounds. In terms of test performance, we find that using more approximate gradients can at the same time improve runtime as well as test accuracy and test log-likelihood.

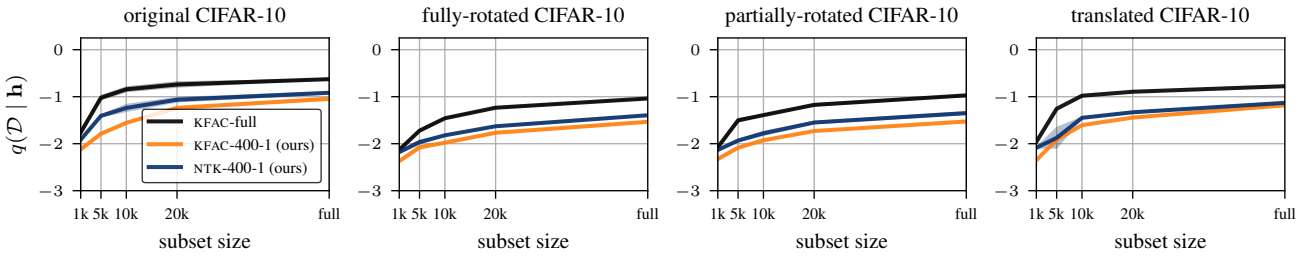

Figure 14. Comparison of marginal likelihoods on different subsets of CIFAR-10 datasets.

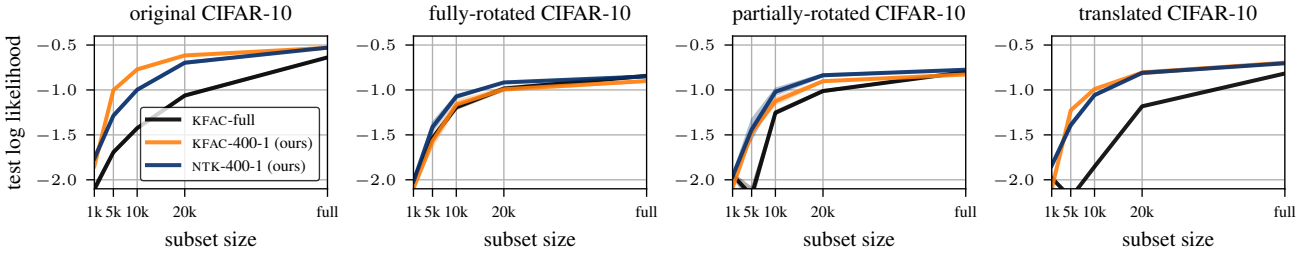

Figure 15. Comparison of test log-likelihood on different subsets of CIFAR-10 datasets.

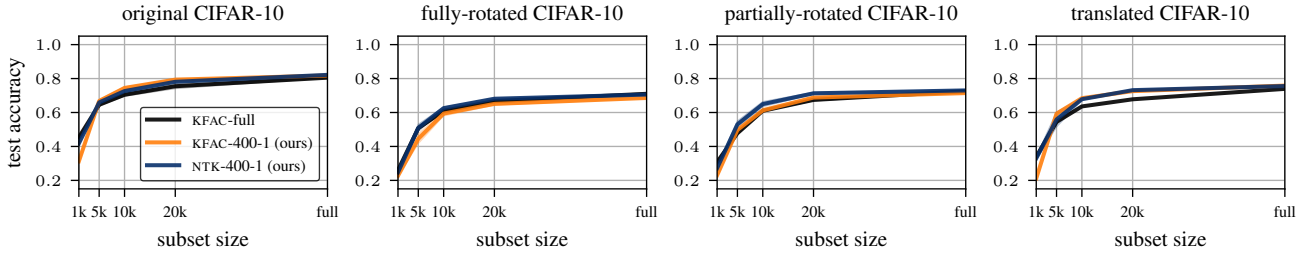

Figure 16. Comparison of test accuracy on different subsets of CIFAR-10 datasets.