Adapting the Linearised Laplace Model Evidence for Modern Deep Learning

Javier Antorán [1] **David Janz** [*2] **James Urquhart Allingham** [*1] **Erik Daxberger** [1 3] **Riccardo Barbano** [4]
Eric Nalisnick [5] **José Miguel Hernández-Lobato** [1]

Abstract

The linearised Laplace method for estimating model uncertainty has received renewed attention in the Bayesian deep learning community. The method provides reliable error bars and admits a closed form expression for the model evidence, allowing for scalable selection of model hyperparameters. In this work, we examine the assumptions behind this method, particularly in conjunction with model selection. We show that these interact poorly with some now-standard tools of deep learning—stochastic approximation methods and normalisation layers—and make recommendations for how to better adapt this classic method to the modern setting. We provide theoretical support for our recommendations and validate them empirically on MLPs, classic CNNs, residual networks with and without normalisation layers, generative autoencoders and transformers.

1. Introduction

Model selection and uncertainty estimation are two important open problems in deep learning. The former aims to select network hyperparameters and architectures without costly cross-validation (Mackay, 1992; Immer et al., 2021a). The latter provides a measure of fidelity of network predictions that can be used in downstream tasks such as experimental design (Barbano et al., 2022b), sequential decision making (Janz et al., 2019) and in safety-critical settings (Fridman et al., 2019). In this paper, we consider a classical approach to these two problems: the linearised Laplace method (Mackay, 1992), which has recently been shown to be one of the best performing methods for inference in neural networks (Khan et al., 2019; Kristiadi et al., 2020; Immer et al., 2021b; Daxberger et al., 2021b;a).

[*]Equal contribution [1]University of Cambridge [2]University of Alberta [3]Max Planck Institute for Intelligent Systems, Tübingen [4] University College London [5]University of Amsterdam. Correspondence to: Javier Antorán <ja666@cam.ac.uk>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

Linearised Laplace approximates the output of a neural network (NN) with a first order Taylor expansion (a linearisation) around optimal NN parameters. It then uses standard linear-model-type error bars to approximate the uncertainty in the output of the NN, while retaining the NN point-estimate as the predictive mean. The latter feature means that, unlike other Bayesian deep learning procedures, the linearised Laplace uncertainty estimates do not come at the cost of the accuracy of the predictive mean (Snoek et al., 2019; Ashukha et al., 2020; Antoran et al., 2020). A downside of the method is that its uncertainty estimates are very sensitive to the choice of certain (prior) hyperparameters (Daxberger et al., 2021b). Our work looks at the model evidence maximisation method for choosing these hyperparameters, as used in the seminal work of Mackay (1992) and later by Immer et al. (2021a). In contrast with often used cross-validation, evidence maximisation reduces model selection to an (often convex) optimisation problem, and can scale to a large number of hyperparameters.

Our contributions are the identification of certain incompatibilities between the assumptions underlying the classical linearised Laplace model evidence and modern deep learning methodology, and a number of recommendations on how to adapt the method in light of these. In particular:

- A core assumption of linearised Laplace is that the point of linearisation is a minimum of the training loss. When the neural network is not trained to convergence (and this is almost never done), this does not hold and results in severe deterioration of the model evidence estimate. We show that this can be corrected by instead considering the optima of the linearised model's loss, that is solving a convex optimisation problem.

- We show that for networks with normalisation layers (such as batch norm (Ioffe & Szegedy, 2015)), the linearised Laplace predictive distribution can fail to be well-defined. However, this can be resolved by separately parametrising the prior corresponding to normalised and non-normalised network parameters.

We provide both theoretical and empirical justification for both points above. The resulting recommended procedure significantly outperforms a naïve implementation on a series of standard tasks and a wide range of neural architectures.

2. The linearised Laplace method

We begin by reviewing a near-textbook version of the linearised Laplace method for the tractable approximation of uncertainty estimates for neural networks, with model evidence maximisation for hyperparamter selection (MacKay, 2003; Bishop, 2006; Immer et al., 2021b;a). We focus on the motivation and assumptions behind this method.

Notation For a vector v and a positive semidefinite matrix M of compatible dimensions, $\|v\|_M$ denotes the norm $\sqrt{v^T M v}$ and we write $\|v\| = \|v\|_I$. We denote by $\cdot$ a vector-matrix or matrix-matrix product where this may help with clarity. For m a positive integer and f a function with parameters p, $\partial_p^m[f(p)](\tilde{p})$ denotes the mth order mixed partial derivatives of f with respect to p evaluated at $\tilde{p}$. We use $\partial_p^m f(\tilde{p})$ where no ambiguity exists. We assume such partial derivatives are well defined pointwise where needed.

Model We consider performing regression or classification using a neural network of the form $\mu(f(\theta, x))$, where $f\colon \Theta \times \mathcal{X} \mapsto \mathcal{Y}$ is a neural network with parameter space Θ, input space $\mathcal{X}$, output space $\mathcal{Y}$ and μ is a linking function. We take $\mathcal{X}, \mathcal{Y}, \Theta$ to be subsets of finite dimensional Euclidean spaces. We assume that we train the network f by minimising a loss function of the form

$$\mathcal{L}_{f,\Lambda}(\theta) = L(f(\theta, \cdot)) + \|\theta\|^2_\Lambda, \tag{1}$$

where $L\colon \mathcal{Y}^{\Theta \times \mathcal{X}} \mapsto \mathbb{R}_+$ gives the data-fit and is of the form $L(f(\theta, \cdot)) = \sum_i \ell(f(\theta, x_i), y_i)$ for some negative log-likelihood function ℓ. We assume L is a proper convex function. The term $\|\theta\|^2_\Lambda$ is a regulariser corresponding to the log density of a Gaussian prior over θ. We take Λ, a positive-definite matrix, to be a model hyperparameter.

The loss minimisation viewpoint set out in (1) is equivalent to finding the *maximum a posteriori* (MAP) solution to a posterior P for θ defined by the Lebesgue density

$$\frac{dP}{d\nu}(\theta) = \frac{1}{Z} e^{-\mathcal{L}_{f,\Lambda}(\theta)}, \tag{2}$$

for $Z > 0$ a normalisation constant. Our aim will be to estimate the posterior distribution for $\mu(f(\theta, x))$ when $\theta \sim P$. For ease of exposition, we focus the posterior for f throughout; the push-forward by μ adds no further complexity.

Approximation The method assumes that we have some parameters $\tilde{\theta} \in \Theta$ obtained by optimising the neural network loss $\mathcal{L}_{f,\Lambda}$ for some initial choice of Λ. For this section, we assume that $\tilde{\theta}$ is a minimum of $\mathcal{L}_{f,\Lambda}$. We proceed to apply approximations to f and P based on consecutive Taylor expansions about $\tilde{\theta}$.

First, we approximate f with the affine function

$$h(\theta, x) = f(\tilde{\theta}, x) + \partial_\theta f(\tilde{\theta}, x) \cdot (\theta - \tilde{\theta}), \tag{3}$$

a first order approximation to f about $\tilde{\theta}$. We then approximate the loss function for the linearised model, $\mathcal{L}_{h,\Lambda}(\theta) = L(h(\theta, \cdot)) + \|\theta\|^2_\Lambda$, with a second order Taylor expansion about $\tilde{\theta}$. Since $\partial_\theta h(\tilde{\theta}, \cdot) = \partial_\theta f(\tilde{\theta}, \cdot)$ and $\tilde{\theta} \in \operatorname{argmin}_\theta \mathcal{L}_{f,\Lambda}$, we have that $\partial_\theta \mathcal{L}_{h,\Lambda}(\tilde{\theta}) = \partial_\theta \mathcal{L}_{f,\Lambda}(\tilde{\theta}) = 0$, and thus the first order term vanishes. This leaves us with the approximation

$$\mathcal{G}_{\tilde{\theta}}(\theta) = \mathcal{L}_{h,\Lambda}(\tilde{\theta}) + \frac{1}{2}\|\theta - \tilde{\theta}\|^2_{\partial^2_\theta \mathcal{L}_{h,\Lambda}(\tilde{\theta})}. \tag{4}$$

We use the approximate loss $\mathcal{G}_{\tilde{\theta}}$ to define an approximate posterior Q by taking its Lebesgue density to be proportional to $e^{-\mathcal{G}_{\tilde{\theta}}(\theta)}$. By inspection, Q is then Gaussian with mean $\tilde{\theta}$ and covariance $(\partial^2_\theta \mathcal{L}_{h,\Lambda}(\tilde{\theta}))^{-1}$. The approximate predictive posterior is given by $h(\theta, \cdot)$, $\theta \sim Q$. Since h is affine, this is again Gaussian. Writing the Hessian of $\mathcal{L}_{h,\Lambda}$ at $\tilde{\theta}$ as

$$\partial^2_\theta \mathcal{L}_{h,\Lambda}(\tilde{\theta}) = H + \Lambda \quad \text{with} \quad H = \partial^2_\theta [L(h(\theta, \cdot))](\tilde{\theta}), \tag{5}$$

and $J(\cdot) = \partial_\theta f(\tilde{\theta}, \cdot)$ for the Jacobian of f at $\tilde{\theta}$, we can write the approximate predictive posterior distribution as

$$h(\theta, \cdot) \sim \mathcal{N}\left(f(\tilde{\theta}, \cdot),\ \|J(\cdot)\|^2_{(H+\Lambda)^{-1}}\right). \tag{6}$$

We have thus augmented the neural network mean with Gaussian error bars obtained from a linear model based on a feature expansion given by the Jacobian of the network.

Model selection The predictive posterior $h(\theta, \cdot)$, $\theta \sim Q$ given in (6) has an explicit dependence on Λ. This parameter significantly affects the predictive posterior variance, but we have no method for choosing it *a priori*. We instead follow an empirical-Bayes procedure: we interpret Λ as the precision of a prior $Q_0 = \mathcal{N}(0, \Lambda)$ and choose Λ as that most likely to generate the observed data given the model $h(\theta, \cdot)$ with $\theta \sim Q_0$. This yields the (log) objective

$$\mathcal{M}_{\tilde{\theta}}(\Lambda) = \log \int_\Theta e^{-\mathcal{G}_{\tilde{\theta}}(\theta)} d\nu, \tag{7}$$

called the model evidence, with ν the Lebesgue measure. Here, $\mathcal{M}_{\tilde{\theta}}(\Lambda)$ depends on Λ implicitly through $\mathcal{G}_{\tilde{\theta}}$. Our approximation $\mathcal{G}_{\tilde{\theta}}$ gives a tractable expression for $\mathcal{M}_{\tilde{\theta}}$,

$$\mathcal{M}_{\tilde{\theta}}(\Lambda) = -\frac{1}{2}\left[\|\tilde{\theta}\|^2_\Lambda + \log\det(\Lambda^{-1} H + I)\right] + C, \tag{8}$$

where C is independent of Λ. Throughout, we will constrain Λ to the set of positive diagonal matrices, as in Mackay (1992). Maximising $\mathcal{M}_{\tilde{\theta}}$ is a concave optimisation problem.

Discussion We made a number of assumptions in our derivation. First, that the data-fit term L is convex. This is satisfied by the standard losses used to train neural networks. We also assumed that the posterior over weights P is sharply peaked around its optima such that it can be approximated well by a quadratic expansion and that h is a good approximation to f near the linearisation point. These assumptions

we do not question further. We made one further important assumption, that the linearisation point $\tilde{\theta}$ is a local minimum of $\mathcal{L}_{f,\Lambda}$ and thus it is also a minima of the linearised loss $\mathcal{L}_{h,\Lambda}$. This final assumption will be the focus of our work.

Our overview of the linearised Laplace method diverged from the original presentation in one significant manner. Mackay (1992) presents the method as an Expectation Maximisation (EM) procedure: in the E step, we perform optimisation to find $\tilde{\theta}$ and compute the approximate posterior Q using this $\tilde{\theta}$ as in (4); in the M step, the prior precision Λ is optimised such that the model evidence is maximised (8). The E and M steps are iterated until convergence. In modern settings, retraining the network (that is, performing the E step more than once) is too expensive. A single step of the EM procedure (as presented) is used instead. The posterior predictive mean is set to match $f(\tilde{\theta}, \cdot)$, ignoring that $\tilde{\theta}$ will no longer be a mode of $\mathcal{L}_{f,\Lambda}$ after an update of Λ. This choice keeps the NN's predictions unchanged, and is considered an advantage of linearised Laplace over competing Bayesian deep learning methods.

3. Overview of results

The linearised Laplace method with model-evidence maximisation procedure, as described, is mostly as first introduced by Mackay (1992). Since then, deep learning training procedures and architectures have changed. Stochastic first order methods are used to minimise the loss function in place of the second order full-batch methods common in classical literature (LeCun et al., 1996; Amari et al., 2000). We often do not use a low value of the loss $\mathcal{L}_{f,\Lambda}$ as a stopping criterion, but instead monitor some separate validation metric. Also, normalisation layers are ubiquitous.

Since the derivations of Section 2 assume that we linearise f and expand $\mathcal{L}_{h,\Lambda}$ about a local minimum of $\mathcal{L}_{f,\Lambda}$ (and thus of $\mathcal{L}_{h,\Lambda}$), modern practises pose difficulties for the presented method. The rest of this section explores these issues in turn, proposes a modern adaptation of the linearised Laplace method, and discusses some interesting special cases.

3.1. An alternative adaptation of linearised Laplace

We consider a naïve implementation of the linearised Laplace method in the context of modern neural networks as that using the linearisation point $\tilde{\theta}$ in the expression for the model evidence $\mathcal{M}_{\tilde{\theta}}$ (8), even if this point is known to not be a local minimum of $\mathcal{L}_{f,\Lambda}$. We now propose an alternative.

We begin, as before, by linearising f about $\tilde{\theta}$, the point returned by (possibly stochastic or incomplete) optimisation of the neural network loss $\mathcal{L}_{f,\Lambda}$ and constructing the feature expansion $J\colon x \mapsto \partial_\theta f(\tilde{\theta}, \cdot)$. Under broad assumptions discussed in Section 3.3, this choice means the posterior mean $f(\tilde{\theta}, \cdot)$ is contained within the linear span of the Jacobian features $J(\cdot)$. This yields credence to the interpretation of the resulting error bars as uncertainty about the NN output.

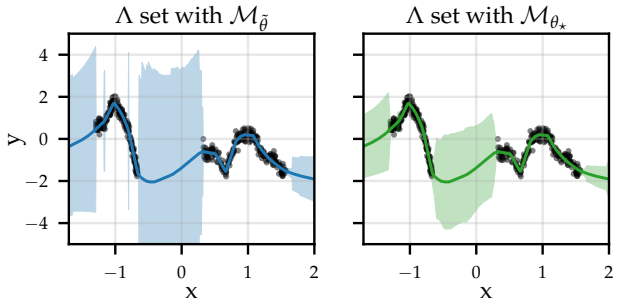

Figure 1. Linearised Laplace predictive mean and std-dev for a 2.6k parameter MLP trained on toy data from Antoran et al. (2020). Choosing Λ with $\mathcal{M}_{\tilde{\theta}}$ yields error bars larger than the marginal std-dev of the targets. Recommendation 1 (using $\mathcal{M}_{\theta_\star}$) solves this.

We diverge from Section 2 in how we approximate $\mathcal{L}_{h,\Lambda}$. We start by noting that since $\tilde{\theta}$ is not a local minimum of $\mathcal{L}_{f,\Lambda}$, it is not one of $\mathcal{L}_{h,\Lambda}$ either (proposition 3). Thus, $\tilde{\theta}$ is not a suitable point for a quadratic approximation to $\mathcal{L}_{h,\Lambda}$. However, for any given Λ, the loss for the linearised model $\mathcal{L}_{h,\Lambda}$ is a convex function of θ (L is convex and h is linear in θ), and thus has a well defined minimiser; expanding the loss about this minimiser will yield a more faithful approximation. Moreover, for each fixed θ, $\mathcal{M}_\theta(\Lambda)$ is concave in Λ, yielding a maximiser. Iteratively minimising the convex $\mathcal{L}_{h,\Lambda}(\theta)$ and maximising the concave $\mathcal{M}_\theta(\Lambda)$ yields a simultaneous stationary point $(\theta_\star, \Lambda_\star)$ satisfying

$$\theta_\star \in \operatorname{argmin}_\theta \mathcal{L}_{h,\Lambda_\star}(\theta) \quad \text{and} \quad \Lambda_\star \in \operatorname{argmax}_\Lambda \mathcal{M}_{\theta_\star}(\Lambda).$$

Our adaption performs evidence maximisation with an affine model h where the basis expansion J is fixed. Unlike Mackay (1992), we do not retrain the neural network.

In practice, we make one further approximation: rather than evaluating the curvature $\partial^2_\theta L(h(\theta, \cdot))$ afresh at successive modes of $\mathcal{L}_{h,\Lambda}$ found during the iterative procedure for computing $(\theta_\star, \Lambda_\star)$, we use the curvature at the linearisation point $H = \partial^2_\theta[L(h(\theta, \cdot))](\tilde{\theta})$ throughout. This avoids the expensive re-computation of the Hessian; experimentally we find that this does not affect the results (Section 5.1). The resulting model evidence expression matches that in (8), with only the weights featuring in the norm changed,

$$\mathcal{M}_{\theta_\star}(\Lambda) = -\frac{1}{2}\left[\|\theta_\star\|^2_\Lambda + \log\det(\Lambda^{-1}H + I)\right] + C. \quad (9)$$

Recommendation 1. *While using the linearisation point $\tilde{\theta}$ in the construction of the feature expansion J and the Hessian H (as introduced in Section 2), find a joint optimum $(\theta_\star, \Lambda_\star)$ for the feature-linear model and employ these to construct the corresponding model evidence $\mathcal{M}_{\theta_\star}$ (equation (9)) and to compute the predictive variance (equation (6)).*

We thus recommend employing a posterior distribution for h of the same form as given in (6), but with prior precision

$\Lambda_\star$. In Section 5, we provide extensive evidence supporting our alternative adaptation of the linearised Laplace model evidence. Figure 1 provides an illustrative example.

3.2. Linearised Laplace with normalised networks

We now study linearised Laplace in the presence of normalisation layers. For this, we put forth the following formalism:

Definition 1 (Normalised networks)**.** We say that a set of networks $\mathcal{F} \subset \mathcal{Y}^{\Theta \times \mathcal{X}}$ is normalised if Θ can be written as a *direct sum* $\Theta' \oplus \Theta''$, with Θ'' non-empty, such that for all networks $f \in \mathcal{F}$ and parameters $\theta' + \theta'' \in \Theta' \oplus \Theta''$,

$$f(\theta' + \theta'', \cdot) = f(\theta' + k\theta'', \cdot)$$

for all $k > 0$.

Throughout, we write θ', θ'' to denote the respective projections onto Θ', Θ'' of a parameter $\theta \in \Theta$; for ease of notation, we will assume these projections are aligned with a standard basis on Θ. That is, we write our parameter vectors as the sum of $k\theta''$, which is only non-zero for normalised weights and f is invariant to k, and θ', for which the opposite holds. We illustrate this with an example in Appendix B.

Our formalism requires only that a single group of normalised parameters Θ'' exists. However, by applying the definition repeatedly, we encompass networks with any number of normalisation layers, and all our results extend to this case. The formalism can be used to model the effects of layer norm (Ba et al., 2016), group norm (Wu & He, 2020) or batch norm (Ioffe & Szegedy, 2015), and even some so-called normalisation-free methods (Brock et al., 2021a;b).

Our focus on normalised networks is motivated by the following observation:

Proposition 1. *For any normalised network f and positive definite matrix Λ, the loss $\mathcal{L}_{f,\Lambda}$ has no local minima.*

To see this, note that the data term fit $L(f(\theta' + k\theta'', \cdot))$ is invariant to the choice of $k > 0$, but we can always decrease the prior term $\|\theta' + k\theta''\|^2_\Lambda$ by decreasing k. Since $k \in \mathbb{R}_+$ has no minimal value, $\mathcal{L}_{f,\Lambda}(\theta' + k\theta'') = L(f(\theta' + k\theta'', \cdot)) + \|\theta' + k\theta''\|^2_\Lambda$ has no local minima.

As in Section 3.1, minimisers of the linear loss $\mathcal{L}_{h,\Lambda}$ remain well-defined (the loss remains strictly convex). However, in this case, $\tilde{\theta}$ cannot minimise $\mathcal{L}_{h,\Lambda}$: the linearisation point minimises $\mathcal{L}_{h,\Lambda}$ only if it minimises $\mathcal{L}_{f,\Lambda}$ (proposition 3). To correct this, from hereon we follow recommendation 1.

An even larger concern raised by proposition 1 is that the linearisation point is identified only up to the scaling k of the normalised parameters θ''. Since k is arbitrary, and does not affect the predictions of the neural network (by definition), it ought not affect the predictive variance returned by the linearised Laplace method. However, in general, it does. See Figure 2 for a demonstration of this. Nonetheless, as shown by the following proposition, it suffices to regularise the normalised parameters θ'' separately from θ' to recover a unique predictive posterior independent of k.

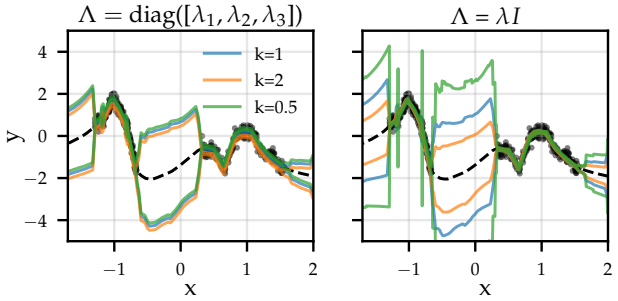

Figure 2. For a normalised MLP with an isotropic prior precision, modifying the scale of the normalised weights $\tilde{\theta}''$ in the linearisation point changes the error bars after hyper-parameter optimisation (right). Incorporating recommendation 2 fixes the issue (left).

Proposition 2. *For normalised neural networks, using a regulariser of the form $\|\theta'\|^2_{\Lambda'} + \|\theta''\|^2_{\Lambda''}$ with Λ' and Λ'' parametrised independently and chosen according to recommendation 1, the predictive posterior $h(\theta, \cdot)$, $\theta \sim Q$ induced by a linearisation point $\theta' + k\theta''$ is independent of the choice of $k > 0$.*

Briefly, the result follows because the Jacobian entries corresponding to weights $k\theta''$ scale with k^{-1}. This is illustrated in Figure 3 (leftmost plot), where as we move further from the origin, weight settings of equal likelihood $L(f(\theta, \cdot))$ move further from each other. Given an un-scaled reference solution $(\theta_\star, \Lambda_\star)$, as we vary k in $k\theta''$, the linear model weights and prior precisions that simultaneously optimise $\mathcal{L}_{h,\Lambda_\star}$ and $\mathcal{M}_{\theta_\star}$ scale as $(\theta'_\star, k\theta''_\star)$, and $(\Lambda'_\star, k^{-2}\Lambda''_\star)$ respectively. These scalings cancel each other in the predictive posterior, which remains invariant. When Λ'' can not change independently of Λ', this cancellation does not occur.

By induction, proposition 2 applies to networks with multiple normalisation layers. Our recommendation is thus:

Recommendation 2. *When using the linearised Laplace method with a normalised network, use an independent regulariser for each normalised parameter group present.*

An example of a suitable regulariser for a network with normalised parameter groups $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(L)}$ and non-normalised parameters θ' would be

$$\lambda' \|\theta'\|^2 + \lambda_1 \|\theta^{(1)}\|^2 + \lambda_2 \|\theta^{(2)}\|^2 + \ldots + \lambda_L \|\theta^{(L)}\|^2$$

for independent parameters $\lambda', \lambda_1, \lambda_2, \ldots, \lambda_L > 0$. In general, this involves setting independent priors for each layer of the network. We note that proposition 2 holds even when $\Lambda_\star$ is found by evaluating the Hessian at the linearisation point $\tilde{\theta}$ instead of $\theta_\star$, as suggested in Section 3.1.

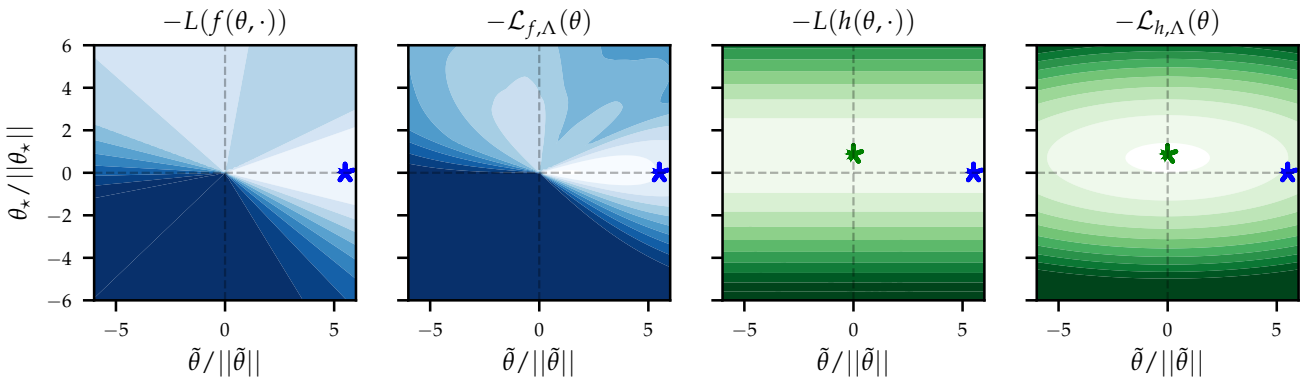

Figure 3. Log likelihood $L(f(\theta,\cdot))$ (left) and log posterior $\mathcal{L}_f$ (left middle) density for an MLP with layer norm, both plotted as functions of a 2d slice of the input layer weights. The horizontal axis corresponds to the direction of $\tilde{\theta}$ while the vertical to $\theta_\star$. The linearization point found with SGD $\tilde{\theta}$ (★) is not an optima of $\mathcal{L}_{f,\Lambda}$ or $\mathcal{L}_{h,\Lambda}$. The linear model log posterior $\mathcal{L}_{h,\Lambda}$ (right) is convex and optimised by $\theta_\star$ (★).

3.3. Networks with a dense final layer

We now look at networks with a dense linear final layer, a (very general) special case which provides further insight into linearised Laplace. These are networks of the form

$$f(\theta,\cdot) = \phi(\theta_{d:},\cdot)\cdot\theta_{:d}, \tag{10}$$

where $\phi(\theta_{d:},\cdot)$ is the output of the penultimate layer and $\theta_{:d}$ are the final layer weights.[1] The derivative of the neural network with respect to the dense final layer weights is

$$\partial_{\theta_{:d}} f(\theta,\cdot) = \phi(\theta_{d:},\cdot),$$

and thus $\phi(\theta_{d:},\cdot)$ is contained within the Jacobian matrix. Consequently, the neural network output $f(\theta,\cdot)$ is always contained in the linear span of the Jacobian basis. This motivates recommendation 1, where we argue for the use of $\tilde{\theta}$ for network linearisation, as it allows for an easy linear model error-bars interpretation for the resulting uncertainty.

Also, the form of the linearised model h simplifies in the dense final layer case when the network is fully normalised. That is $f(\theta'+\theta'',\cdot) = \phi(\theta''_{d:},\cdot)\cdot\theta'_{:d}$. Here, the derivative of ϕ in the direction of $\tilde{\theta}''_{d:}$ is zero (see Appendix C.2):

$$\partial_{\theta''_{d:}}\phi(\tilde{\theta}''_{d:},\cdot)\cdot\tilde{\theta}''_{d:} = 0. \tag{11}$$

Thus cancellation occurs in (3), yielding a model of the form

$$h(\theta,\cdot) = \partial_\theta f(\tilde{\theta},\cdot)\cdot\theta, \quad \theta\sim Q. \tag{12}$$

That is, a linear model based on the features $J(\cdot)=\partial_\theta f(\tilde{\theta},\cdot)$.

3.4. Further implications and discussion

Here, and in Appendix E, we discuss details and implications of the presented recommendations and results.

Finding the optimum of the linear model Our adapted linearised Laplace method requires identifying the joint stationary point $(\theta_\star,\Lambda_\star)$. In general, this does not admit a closed-form solution. Instead, we alternate gradient-based optimisation of $\mathcal{L}_{h,\Lambda}$ and $\mathcal{M}_\theta$. In Appendix D, we provide a derivation for the gradient of $\mathcal{L}_{h,\Lambda}$, algorithm 1 for its computation and discuss implementation trade-offs. For normalised networks with dense output layers, implementing the simplified linear model (12) directly yields faster and more stable optimisation. Obtaining the gradients of $\mathcal{M}_\theta$ involves computing Hessian log-determinants, which in turn requires approximations in the context of large networks.

Magnitude of linearisation point in normalised networks Optimising a normalised neural network returns a solution for the normalised weights (those in Θ'') up to some scaling factor $k>0$. How is k determined? Recall, from (11), that for any $\theta''\in\Theta''$, the directional derivative of the NN output in the direction of θ'' is zero. This is also illustrated in Figure 3. With this in mind, the dynamics of optimisation can be understood by analogy to a Newtonian system in polar coordinates. The weights are a mass upon which the data fit gradient acts as a tangential force. When discretised, this gradient pushes the weights away from zero. On the other hand, regularisation from the prior term acts like a centripetal force, pushing the weights towards the origin. The resulting k is thus proportional to the variance of the gradients of θ'', and as such dependent on the learning rate and batch size hyperparameters, while being inversely proportional to the regularisation strength, e.g. weight decay. This has been studied extensively in the optimisation literature, including van Laarhoven (2017); Hoffer et al. (2018); Cai et al. (2019); Li et al. (2020); Lobacheva et al. (2021).

On network biases in the Jacobian feature expansion Most normalisation techniques introduce scale invariance by dividing subsets of network activations by an empirical estimate of their standard deviation. These activations depend on the values of both weights and biases. On the other hand, practical use of linearised Laplace commonly considers uncertainty due to only network weights (Daxberger et al., 2021b; Maddox et al., 2021), excluding bias entries

[1] The subindex d: in $\theta_{d:}$ excludes last layer weights from θ.

from Jacobian and Hessian matrices. This departure from our assumptions can break the scale invariance necessary for lemma 4. Whether invariance is preserved for the weights in the bias-exclusion setting depends on the relative effect of weights and biases on each subset of normalised activations. Invariance is preserved if the biases have small impact. Empirically, we find that the inclusion (or exclusion) of biases does not alter the improvements obtained from applying our recommendations (see Figure 4).

Implications for the (non-linearised) Laplace method The (non-linearised) Laplace method (Ritter et al., 2018; Kristiadi et al., 2020) approximates the intractable posterior P by means of a quadratic expansion around an optima, but without the linearisation step given in equation (3). As discussed in Section 3.1, when employing stochastic optimisation, early stopping, or normalisation layers, we will not find a minimiser of $\mathcal{L}_{f,\Lambda}$. Without a well-behaved surrogate linear model loss to fall back on, the Laplace method can yield very biased estimates of the model evidence.

4. Formal results and proofs

Proposition 3. *For network f with linearisation h about $\tilde{\theta}$ and a positive definite regulariser Λ, if $\tilde{\theta}$ is not a stationary point of $\mathcal{L}_{f,\Lambda}$, it is not a local minimum of $\mathcal{L}_{h,\Lambda}$.*

Proof. Since $\tilde{\theta}$ is not a stationary point of $\mathcal{L}_{f,\Lambda}$, the gradient $\partial_\theta \mathcal{L}_{f,\Lambda}(\tilde{\theta})$ is not identically zero. But $\partial_\theta \mathcal{L}_{f,\Lambda}(\tilde{\theta})$ equal to

$$\sum_i \partial_f \ell(f(\tilde{\theta}, x_i), y_i) \cdot \partial_\theta f(\tilde{\theta}, x_i) + \partial_\theta[\|\theta\|^2_\Lambda](\tilde{\theta}) = \sum_i \partial_h \ell(h(\tilde{\theta}, x_i), y_i) \cdot \partial_\theta h(\tilde{\theta}, x_i) + \partial_\theta[\|\theta\|^2_\Lambda](\tilde{\theta}),$$

which is in turn equal to $\partial_\theta \mathcal{L}_{h,\Lambda}(\tilde{\theta})$. Since this is thus non-zero, $\tilde{\theta}$ cannot be a local minimum of $\mathcal{L}_{h,\Lambda}$. □

The proof of proposition 1 was sketched in text; we omit it.

For the proof of proposition 2, we use the following notation. Consider a linearisation point $\theta' + \theta''$, with corresponding linearised function h, basis function J and Hessian H. For $k > 0$, h_k, J_k, H_k denote these quantities corresponding to a linearisation point $\theta_k := \theta' + k\theta''$. Moreover, we write

$$J = \begin{bmatrix} J' \\ J'' \end{bmatrix} \quad \text{and} \quad H = \begin{bmatrix} H' & X^T \\ X & H'' \end{bmatrix}$$

for the sub-entries of J and H with dependencies on θ' and θ'' respectively, with X containing cross-terms. We refer to sub-entries of J_k and H_k in the same manner.

With notation in place, we have the following scaling result:

Lemma 4. *Let f be a normalised network and consider two alternative linearisation points $\tilde{\theta}' + \tilde{\theta}''$ and $\tilde{\theta}' + k\tilde{\theta}''$ for some $k > 0$. Then,*

$$\begin{bmatrix} J'_k \\ k^{-1}J''_k \end{bmatrix} = J \quad \textit{and} \quad \begin{bmatrix} H'_k & k^{-1}X_k^T \\ k^{-1}X_k & k^{-2}H''_k \end{bmatrix} = H.$$

Moreover, for all $\theta \in \Theta$, $h_k(\theta' + k\theta'', \cdot) = h(\theta' + \theta'', \cdot)$.

The proof of the above lemma is presented in Appendix C.1. We now use it to show how the optimal weights and regularisation parameters scale with the parameter k.

Lemma 5. *For $k > 0$, let h_k be a linearisation of a normalised network f about $\tilde{\theta}' + k\tilde{\theta}''$. Then (θ_k, Λ_k) are an optima of the resulting objectives $(\mathcal{L}_{h_k,\Lambda_k}, \mathcal{M}_{\theta_k})$ respectively if and only if they are of the form*

$$(\theta_k, \Lambda_k) = (\theta'_\star + k\theta''_\star, \Lambda'_\star + k^{-2}\Lambda''_\star)$$

where $(\theta_\star, \Lambda_\star)$ are optima of $(\mathcal{L}_{h,\Lambda_\star}, \mathcal{M}_{\theta_\star})$ with h a linearisation of f about $\tilde{\theta}' + \tilde{\theta}''$.

Proof. To prove the result, we will show that $\mathcal{L}_{h_k,\Lambda_k}(\theta' + k\theta'') = \mathcal{L}_{h,\Lambda_\star}(\theta' + \theta'')$ for all $\theta' + \theta'' \in \Theta$ and $\mathcal{M}_{\theta_k}(\Lambda' + k^{-2}\Lambda'') = \mathcal{M}_{\theta_\star}(\Lambda' + \Lambda'')$ for all strictly diagonal positive matrices Λ', Λ'' of compatible sizes. Then, the result follows by noting that for $k > 0$ fixed, the mappings $\theta' + \theta'' \mapsto \theta' + k\theta''$ and $\Lambda' + \Lambda'' \mapsto \Lambda' + k^{-2}\Lambda''$ are bijections.

Consider the objective $\mathcal{L}_{h_k,\Lambda_k}$. By definition, $\mathcal{L}_{h_k,\Lambda_k}(\theta' + k\theta'')$ is given by

$$L(h_k(\theta' + k\theta'', \cdot)) + \|\theta'\|^2_{\Lambda'_k} + \|k\theta''\|^2_{\Lambda''_k} = L(h(\theta' + \theta'', \cdot)) + \|\theta'\|^2_{\Lambda'_\star} + \|\theta''\|^2_{\Lambda''_\star},$$

where the equality follows by lemma 4 and the definition of Λ_k. The right hand side is equal to $\mathcal{L}_{h,\Lambda_\star}(\theta' + \theta'')$ proving the first equality.

Consider the objective $\mathcal{M}_{\theta_k}$. For our claim, we need to show that

$$\|\theta_k\|_{\Lambda' + k^{-2}\Lambda''} + \log \frac{\det(H_k + \Lambda' + k^{-2}\Lambda'')}{\det(\Lambda' + k^{-2}\Lambda'')} = \|\theta_\star\|_{\Lambda' + \Lambda''} + \log \frac{\det(H + \Lambda' + \Lambda'')}{\det(\Lambda' + \Lambda'')}.$$

The equality $\|\theta_k\|_{\Lambda' + k^{-2}\Lambda''} = \|\theta_\star\|_{\Lambda' + \Lambda''}$ holds trivially. We now show equality of the determinants. Let d, D denote the dimensions of Θ' and Θ'' respectively. By the Schur determinant lemma, the numerator $\det(H_k + \Lambda' + k^{-2}\Lambda'')$ is equal to

$$\det(H''_k + \frac{\Lambda''_{d:}}{k^2}) \det(H'_k + \Lambda'_{:d} - X(H''_k + \frac{\Lambda''_{d:}}{k^2})^{-1}X^T),$$

where $\Lambda'_{:d} = [\Lambda'_{ij} : i, j \leq d]$ and $\Lambda''_{d:}$ is defined similarly. Using lemma 4, $\det(H''_k + \frac{\Lambda''_{d:}}{k^2}) = (\frac{1}{k^2})^D \det(H'' + \Lambda''_{d:})$. Expanding the Schur complement term and using lemma 4 shows that it is independent of k. In turn, the denominator is given by

$$\begin{aligned}\det(\Lambda' + k^{-2}\Lambda'') &= (\frac{1}{k^2})^D \det(\Lambda'_{:d}) \det(\Lambda''_{d:}) \\ &= (\frac{1}{k^2})^D \det(\Lambda' + \Lambda''),\end{aligned}$$

The $(\frac{1}{k^2})^D$ terms in the numerator and denominator cancel, yielding the claim. □

Proof of proposition 2. Using lemma 4 and lemma 5 and the notation defined therein,

$$\|J\|_{(H+\Lambda_\star)^{-1}} = \|J_k\|_{(H_k+\Lambda_k)^{-1}}.$$

Thus the errorbars induced by linearising about $\tilde{\theta}' + \tilde{\theta}''$ and $\tilde{\theta}' + k\tilde{\theta}''$ are equal for all $k > 0$. □

Note that the proof of the results required for proposition 2 depends crucially on being able to scale Λ''_k with k while keeping Λ'_k fixed. This motivates recommendation 2.

5. Experiments

We proceed to provide empirical evidence for our assumptions and recommendations. Specifically, in Section 5.1, we validate the assumptions made in Section 3. Then, in Section 5.2, we demonstrate that our recommendations yield improvements across a wide range of architectures. In these first two subsections, we employ networks containing at most 46k weights, since this is the largest model for which we can tractably compute the Hessian on an A100 GPU. This choice avoids confounding the effects described in Section 3 with any further approximations. In Section 5.3, we show that our recommendations yield performance improvements on the 23M parameter ResNet-50 network while employing the standard KFAC approximation to the Hessian (Martens & Grosse, 2015; Daxberger et al., 2021a).

Unless specified otherwise, we: 1. train a NN to find $\tilde{\theta}$ using standard stochastic optimisation algorithms. 2. linearise the network about $\tilde{\theta}$ as in (3). 3. optimise the linear model weights using $\mathcal{L}_{h,\Lambda}$ (algorithm 1) and layer-wise regularisation parameters with $\mathcal{M}_{\theta_\star}$ (9). 4. compute the posterior predictive distribution as in (6). We repeat this procedure with 5 random seeds and report mean results and standard error. For each seed, the methods compared produce the same mean predictions $f(\tilde{\theta}, \cdot)$, only differing in their predictive variance. In this setting, the test Negative Log-Likelihood (NLL, lower is better) can be understood as a measure of uncertainty miscalibration. The full details of the setup used for each experiment are provided in Appendix F.

5.1. Validation of modelling assumptions

We validate the key conjectures stated in Section 3. If not specified otherwise, we employ a 46k parameter ResNet (He et al., 2016a) with batch-normalisation after every convolutional layer. The output layer is dense, satisfying (12).

Choice of Hessian In Section 3.1, we suggest evaluating the Hessian of $\mathcal{L}_h$ at the linearisation point $\tilde{\theta}$ (instead of $\theta_\star$) for model evidence optimisation (9). This avoids the need to recompute the Hessian throughout optimisation. Figure 4 (left) shows how the improvement from using the recommended model evidence $\mathcal{M}_{\theta_\star}$, as opposed to $\mathcal{M}_{\tilde{\theta}}$, dominates the effect of the choice of Hessian evaluation point.

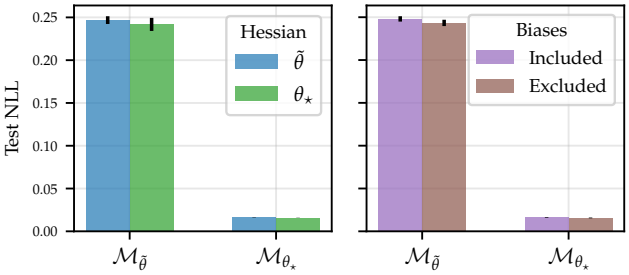

Figure 4. Comparison of the test NLL improvement obtained when switching from $\mathcal{M}_{\tilde{\theta}}$ to $\mathcal{M}_{\theta_\star}$ to optimise the prior precision Λ relative to the impact of (*left*) evaluating the Hessian at $\tilde{\theta}$ or $\theta_\star$, and (*right*) excluding network biases from the basis functions. Both plots use a 46k ResNet with batch norm trained on MNIST.

Dependence on k for isotropic precisions In Figure 2, we illustrate the dependence of the predictive posterior on the scale of normalised weights k for an isotropic prior precision $\Lambda = \lambda I$, i.e. recommendation 2 is ignored. We use a 2.6k parameter 2 hidden layer fully connected NN with layer norm after every layer except the last and a 1d regression task. Changing k changes the optimal λ and, consequently, the predictive uncertainty changes. With layer-wise λ this effect vanishes (as predicted by proposition 2).

Treatment of NN biases Excluding the Jacobians of network biases from our basis function expansion breaks the scaling properties presented in lemma 4. In Figure 4 (right), we show that the effect of excluding biases is dominated by the choice of the model evidence between $\mathcal{M}_{\theta_\star}$ and $\mathcal{M}_{\tilde{\theta}}$.

Early stopping We evaluate whether more thorough optimisation of the NN weights with $\mathcal{L}_f$ leads to a linearisation point $\tilde{\theta}$ closer to $\theta_\star$ in the sense of the implied optimal regularisation and induced posterior predictive distribution. We perform this analysis on normalised and unnormalised networks (which use the non scale-invariant FixUp regularisation instead (Zhang et al., 2019)), since $\tilde{\theta}$ is guaranteed to never match $\theta_\star$ for the former. Surprisingly, the Wasserstein-2 distance between predictive distributions obtained with $\mathcal{M}_{\tilde{\theta}}$ and $\mathcal{M}_{\theta_\star}$ increases with more optimisation steps in both cases. Thus, more thorough optimisation does not help.

Table 1. Validation of recommendations across architectures. All results are reported as negative log-likelihoods (lower is better). In each column, the best performing method is bolded. For each $\mathcal{M}$, if single or layerwise λ optimisation performs better, it is underlined.

		TRANSFORMER	CNN	RESNET	PRE-RESNET	FIXUP	U-NET
$\mathcal{M}_{\theta_\star}$	single λ	**0.162** ± 0.042	**0.025** ± 0.000	**0.017** ± 0.000	0.017 ± 0.000	**0.055** ± 0.006	-1.793 ± 0.050
	layerwise λ	**0.162** ± 0.042	**0.025** ± 0.000	**0.016** ± 0.001	**0.016** ± 0.000	0.061 ± 0.005	**-2.240** ± 0.027
$\mathcal{M}_{\tilde{\theta}}$	single λ	0.310 ± 0.060	0.253 ± 0.001	0.252 ± 0.006	0.220 ± 0.004	0.153 ± 0.021	-1.164 ± 0.052
	layerwise λ	**0.162** ± 0.042	0.205 ± 0.002	0.236 ± 0.005	0.239 ± 0.004	0.200 ± 0.018	-1.703 ± 0.023

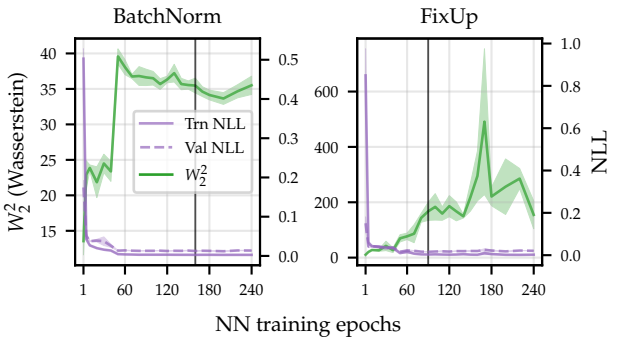

Figure 5. Wasserstein distance between predictive posteriors obtained when using $\mathcal{M}_{\tilde{\theta}}$ and $\mathcal{M}_{\theta_\star}$ at multiple NN training stages. The vertical black line indicates optimal (val-based) early stopping.

5.2. Validation of recommendations across architectures

We evaluate the utility of recommendation 1, and recommendation 2 on a range of architectures and tasks: 1. a transformer architecture on the pointcloud-MNIST variable length sequence classification task. This model uses layer norm in alternating layers. 2. a LeNet-style CNN with batch norm placed after every convolutional layer and a dense output layer tasked with MNIST classification. 3. a ResNet with batch norm after every layer except the dense output layer (MNIST classification). 4. the same ResNet but with batch norm substituted by (non scale-invariant) FixUp regularisation (MNIST classification). 5. a pre-ResNet (He et al., 2016b). This architecture differs from ResNet in that batch norm is placed before each weight layer instead of after them; the implication is that there is only 1 normalised group of weights encompassing all weights but those of the dense output layer (MNIST classification). 6. a fully convolutional U-net autoencoder tasked with tomographic reconstruction (regression) of a KMNIST character from a noisy low-dimensional observation. We reproduce the setting of Barbano et al. (2022a). Group norm is placed after every layer except the last, which is convolutional.

As shown in Table 1, the application of recommendation 1 yields notably improved performance across all settings. Applying recommendation 2 yields modest improvements for classification networks with normalisation layers but large improvements for the U-net. Interestingly, layer-wise regularisation degrades performance in the FixUp ResNet.

Table 2. Test negative log-likelihoods for ResNet-50 on CIFAR10.

		BATCH NORM	FIXUP
$\mathcal{M}_{\theta_{\star,\text{simple}}}$	single λ	**0.112** ± 0.004	0.128 ± 0.000
	layerwise λ	**0.109** ± 0.003	**0.096** ± 0.000
$\mathcal{M}_{\theta_\star}$	single λ	0.190 ± 0.005	0.249 ± 0.002
	layerwise λ	0.194 ± 0.009	0.193 ± 0.001
$\mathcal{M}_{\tilde{\theta}}$	single λ	0.570 ± 0.004	0.412 ± 0.000
	layerwise λ	0.567 ± 0.004	0.360 ± 0.000

5.3. Large scale models

We validate our recommendations on the 23M parameter ResNet-50 network trained on the CIFAR10 dataset. This model places batch norm after every layer except the dense output layer. We also consider a normalisation-free FixUp ResNet-50. Table 2 shows that both of our recommendations yield better test NLL, with larger gains obtained by the batch norm network. The normalisation-free FixUp setting does not simplify as in eq. (12). Nonetheless, assuming the simplified model evidence, denoted $\mathcal{M}_{\theta_{\star,\text{simple}}}$, when obtaining the linear model optima yields improved performance for all models. We expand on this in Appendix G.4.

6. Discussion

We have identified and addressed two pitfalls of a naïve application of linearised Laplace to modern NNs. First, the optima of the loss function is not found in practice. This invalidates the assumption that the point at which we linearise our model is stationary. However, every linearisation point implies an associated basis function linear model. As we use this model to provide errobars, we propose to choose hyperparameters using the evidence of this model. This requires only the solving of a convex optimisation problem, one much simpler than NN optimisation. Second, normalisation layers introduce an invariance to the scale of NN weights and thus the linearisation point can only be identified up to a scaling factor. We show that to obtain a predictive posterior that is invariant to this scaling factor, the regulariser must be independently parametrised for each normalised group of weights, e.g. different layers. Our experiments confirm the effectiveness of these recommendations across a wide range of model architectures and sizes.

Acknowledgements

The authors would like to thank Marine Schimel for helpful discussions. JA acknowledges support from Microsoft Research, through its PhD Scholarship Programme, and from the EPSRC. RB acknowledges support from the i4health PhD studentship (UK EPSRC EP/S021930/1), and from The Alan Turing Institute (UK EPSRC EP/N510129/1). ED acknowledges funding from the EPSRC and Qualcomm. JUA acknowledges funding from the EPSRC, the Michael E. Fisher Studentship in Machine Learning, and the Qualcomm Inovation Felowship. This work has been performed using resources provided by the Cambridge Tier-2 system operated by the University of Cambridge Research Computing Service (http://www.hpc.cam.ac.uk) funded by EPSRC Tier-2 capital grant EP/T022159/1.

References

Amari, S., Park, H., and Fukumizu, K. Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural Comput.*, 12(6):1399–1409, 2000. doi: 10.1162/089976600300015420.

Andrei, N. Accelerated conjugate gradient algorithm with finite difference hessian/vector product approximation for unconstrained optimization. *Journal of Computational and Applied Mathematics*, 230(2):570–582, 2009. ISSN 0377-0427. doi: https://doi.org/10.1016/j.cam.2008.12.024.

Antoran, J. and Miguel, A. Disentangling and learning robust representations with natural clustering. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 694–699, 2019.

Antoran, J., Allingham, J., and Hernández-Lobato, J. M. Depth uncertainty in neural networks. In , *Advances in Neural Information Processing Systems*, volume 33, pp. 10620–10634. Curran Associates, Inc., 2020.

Antorán, J., Bhatt, U., Adel, T., Weller, A., and Hernández-Lobato, J. M. Getting a CLUE: A method for explaining uncertainty estimates. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Antorán, J., Barbano, R., Leuschner, J., Hernández-Lobato, J. M., and Jin, B. A probabilistic deep image prior for computational tomography. *CoRR*, abs/2203.00479, 2022. doi: 10.48550/arXiv.2203.00479.

Ashukha, A., Lyzhov, A., Molchanov, D., and Vetrov, D. P. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Ba, L. J., Kiros, J. R., and Hinton, G. E. Layer normalization. *CoRR*, abs/1607.06450, 2016.

Barbano, R., Antoran, J., Hernández-Lobato, J. M., and Jin, B. A probabilistic deep image prior over image space. In *Fourth Symposium on Advances in Approximate Bayesian Inference*, 2022a.

Barbano, R., Leuschner, J., Antoran, J., Hernández-Lobato, J. M., and Jin, B. Bayesian experimental design for computed tomography with the linearised deep image prior. *ICML Workshop on Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2022b.

Betancourt, M. The fundamental incompatibility of scalable hamiltonian monte carlo and naive data subsampling. In , *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 533–540. JMLR.org, 2015.

Bhatt, U., Antorán, J., Zhang, Y., Liao, Q. V., Sattigeri, P., Fogliato, R., Melançon, G. G., Krishnan, R., Stanley, J., Tickoo, O., Nachman, L., Chunara, R., Srikumar, M., Weller, A., and Xiang, A. Uncertainty as a form of transparency: Measuring, communicating, and using uncertainty. In , *AIES '21: AAAI/ACM Conference on AI, Ethics, and Society, Virtual Event, USA, May 19-21, 2021*, pp. 401–413. ACM, 2021. doi: 10.1145/3461702.3462571.

Bishop, C. M. *Pattern recognition and machine learning*. springer, 2006.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. In , *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1613–1622. JMLR.org, 2015.

Botev, A., Ritter, H., and Barber, D. Practical Gauss-Newton optimisation for deep learning. In , *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 557–565. PMLR, 06–11 Aug 2017.

Brock, A., De, S., and Smith, S. L. Characterizing signal propagation to close the performance gap in unnormalized resnets. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a.

Brock, A., De, S., Smith, S. L., and Simonyan, K. High-performance large-scale image recognition without normalization. In , *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24*

July 2021, Virtual Event, volume 139 of *Proceedings of Machine Learning Research*, pp. 1059–1071. PMLR, 2021b.

Cai, Y., Li, Q., and Shen, Z. A quantitative analysis of the effect of batch normalization on gradient descent. In , *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 882–890. PMLR, 2019.

Chambolle, A. An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1):89–97, 2004.

Chen, T., Fox, E. B., and Guestrin, C. Stochastic gradient hamiltonian monte carlo. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 1683–1691. JMLR.org, 2014.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. Deep learning for classical japanese literature, 2018.

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., and Hennig, P. Laplace redux–effortless Bayesian deep learning. In , *Advances in Neural Information Processing Systems*, volume 34, 6–14 Dec 2021a.

Daxberger, E., Nalisnick, E., Allingham, J. U., Antoran, J., and Hernandez-Lobato, J. M. Bayesian deep learning via subnetwork inference. In , *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2510–2521. PMLR, 18–24 Jul 2021b.

Dusenberry, M. W., Jerfel, G., Wen, Y., Ma, Y.-A., Snoek, J., Heller, K., Lakshminarayanan, B., and Tran, D. Efficient and scalable bayesian neural nets with rank-1 factors. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Foong, A. Y. K., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. 'In-between' uncertainty in Bayesian neural networks. *CoRR*, abs/1906.11537, 2019.

Fridman, L., Ding, L., Jenik, B., and Reimer, B. Arguing machines: Human supervision of black box ai systems that make life-critical decisions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

Graves, A. Practical variational inference for neural networks. In , *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pp. 2348–2356. 2011.

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

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In , *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pp. 630–645. Springer, 2016b. doi: 10.1007/978-3-319-46493-0_38.

Hernández-Lobato, J. M. and Adams, R. P. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In , *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1861–1869. JMLR.org, 2015.

Hernández-Lobato, J. M., Li, Y., Rowland, M., Bui, T. D., Hernández-Lobato, D., and Turner, R. E. Black-box alpha divergence minimization. In , *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1511–1520. JMLR.org, 2016.

Hinton, G. E. and van Camp, D. Keeping the neural networks simple by minimizing the description length of the weights. In , *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, COLT 1993, Santa Cruz, CA, USA, July 26-28, 1993*, pp. 5–13. ACM, 1993.

Hoffer, E., Banner, R., Golan, I., and Soudry, D. Norm matters: efficient and accurate normalization schemes in deep networks. In , *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 2164–2174, 2018.

Immer, A., Bauer, M., Fortuin, V., Rätsch, G., and Khan, M. E. Scalable marginal likelihood estimation for model selection in deep learning. In , *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4563–4573. PMLR, 2021a.

Immer, A., Korzepa, M., and Bauer, M. Improving predictions of bayesian neural nets via local linearization. In

, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pp. 703–711. PMLR, 2021b.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In , *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456. JMLR.org, 2015.

Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. G. What are bayesian neural network posteriors really like? In , *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4629–4640. PMLR, 18–24 Jul 2021.

Janz, D., Hron, J., Mazur, P., Hofmann, K., Hernández-Lobato, J. M., and Tschiatschek, S. Successor uncertainties: Exploration and uncertainty in temporal difference learning. In , *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 4509–4518, 2019.

Khan, M. E., Immer, A., Abedi, E., and Korzepa, M. Approximate inference turns deep networks into gaussian processes. In , *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 3088–3098, 2019.

Kristiadi, A., Hein, M., and Hennig, P. Being bayesian, even just a bit, fixes overconfidence in relu networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5436–5446. PMLR, 2020.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017.

Lawrence, N. D. *Variational inference in probabilistic models*. PhD thesis, University of Cambridge, 2000.

LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient backprop. In *Neural Networks: Tricks of the Trade*, 1996.

Li, Z., Lyu, K., and Arora, S. Reconciling modern deep learning with traditional optimization analyses: The intrinsic learning rate. In , *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Lobacheva, E., Kodryan, M., Chirkova, N., Malinin, A., and Vetrov, D. P. On the periodic behavior of neural network training with batch normalization and weight decay. In , *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 21545–21556, 2021.

Mackay, D. J. C. *Bayesian Methods for Adaptive Models*. PhD thesis, USA, 1992.

MacKay, D. J. C. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003. ISBN 978-0-521-64298-9.

Maddox, W., Tang, S., Moreno, P. G., Wilson, A. G., and Damianou, A. Fast adaptation with linearized neural networks. In , *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pp. 2737–2745. PMLR, 2021.

Maddox, W. J., Benton, G. W., and Wilson, A. G. Rethinking parameter counting in deep models: Effective dimensionality revisited. *CoRR*, abs/2003.02139, 2020.

Martens, J. and Grosse, R. B. Optimizing neural networks with kronecker-factored approximate curvature. In , *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 2408–2417. JMLR.org, 2015.

Nabarro, S., Ganev, S., Garriga-Alonso, A., Fortuin, V., van der Wilk, M., and Aitchison, L. Data augmentation in bayesian neural networks and the cold posterior effect. *CoRR*, abs/2106.05586, 2021.

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

Neal, R. M. Annealed importance sampling. *Stat. Comput.*, 11(2):125–139, 2001. doi: 10.1023/A:1008923215028.

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.

Osawa, K., Swaroop, S., Khan, M. E., Jain, A., Eschenhagen, R., Turner, R. E., and Yokota, R. Practical deep learning with bayesian principles. In , *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019,*

NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp. 4289–4301, 2019.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006. ISBN 026218253X.

Ritter, H., Botev, A., and Barber, D. A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Snoek, J., Ovadia, Y., Fertig, E., Lakshminarayanan, B., Nowozin, S., Sculley, D., Dillon, J., Ren, J., and Nado, Z. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pp. 13969–13980, 2019.

Tomczak, M., Swaroop, S., Foong, A., and Turner, R. Collapsed variational bounds for bayesian neural networks. In , *Advances in Neural Information Processing Systems*, volume 34, pp. 25412–25426. Curran Associates, Inc., 2021.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. S. Deep image prior. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 9446–9454. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00984.

van Laarhoven, T. L2 regularization versus batch and weight normalization. *CoRR*, abs/1706.05350, 2017.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In , *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pp. 681–688. Omnipress, 2011.

Wenzel, F., Roth, K., Veeling, B. S., Swiatkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. How good is the bayes posterior in deep neural networks really? In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10248–10259. PMLR, 2020.

Wu, Y. and He, K. Group normalization. *Int. J. Comput. Vis.*, 128(3):742–755, 2020. doi: 10.1007/s11263-019-01198-w.

Zhang, H., Dauphin, Y. N., and Ma, T. Fixup initialization: Residual learning without normalization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

A. Additional discussion of related work

We commence by providing an overview of previous work on the Laplace approximation for neural networks while discussing its relation to the present paper. We then delve into alternative approximate Bayesian inference methods for neural networks.

A.1. The linearised Laplace approximation for neural networks

The seminal work on Bayesian inference for NNs of Mackay (1992) approximated the intractable posterior over the neural network weight space using the Laplace approximation. The so-called "Evidence Framework" laid out by Mackay (1992) consists of an EM-style iterative algorithm where a NN is fit and Laplace approximated (E-step), and then this posterior approximation is used to update hyperparameters (M-step) such that the Laplace marginal likelihood estimate is maximised. The author further employed a local linearisation of the NN to obtain a closed form approximation of the predictive posterior. However, Mackay (1992) estimated the posterior's curvature using the full Hessian, while mentioning that the Generalised Gauss Newton matrix (GGN), which corresponds to the curvature of the surrogate linear model's loss, is a computationally cheaper alternative. Since then, the GGN approximation has become ubiquitous (Bishop, 2006).

The local linearisation employed by Mackay (1992) for estimating the predictive posterior was seen as an approximation chosen for convenience. With this in mind, Lawrence (2000) forgoed the linearisation step and used Monte Carlo sampling to marginalise the distribution over network weights. This resulted in very poor performance. Ritter et al. (2018) also employed sampling in their work. Similarly to Lawrence (2000), these authors found that the Laplace approximation can place large amounts of probability mass in low density regions of the posterior, yielding poor results. They circumvented this issues by damping their posterior covariance estimate, making it closer to a point mass at the mode.

With the proliferation of larger neural networks with higher dimensional weight spaces, computation of the Hessian became intractably expensive and the Laplace approximation fell out of favour relative to other approximate inference methods discussed in Appendix A.2. The work of Ritter et al. (2018) was the first to break this trend by proposing the use of the the Kronecker factorised (KFAC) approximation (Martens & Grosse, 2015; Botev et al., 2017) to approximate the NN Generalised Gauss Newton (GGN) curvature matrix, making Laplace's method scalable to modern architectures.

Building upon the above work, Khan et al. (2019) made the observation that the GGN posterior precision approximation coincides with the exact posterior of the locally linearised neural network, when seen as a model in its own right. Khan et al. (2019) also suggested the use of the marginal likelihood of this surrogate linear model as a proxy for that of the neural network. Following this, Foong et al. (2019); Immer et al. (2021b) made the observation that assuming linearisation at prediction time significantly improves performance. The latter work (Immer et al., 2021b) expanded on this topic by discussing the view of the GGN Laplace approximation as a model switch into a generalised linear model. The authors note that the linearisation point may not be the optima of the NN loss in practise and that the Laplace distribution does not yield the exact posterior for the linear model when employing non-Gaussian likelihoods e.g. in the classification setting. Motivated by this Immer et al. (2021b), proposed to "refine" the linear model's posterior, which they did using variational inference. This provided some improvements in terms of predictive performance. Our recommendation 1 can be interpreted as suggesting a form of posterior refinement where the posterior covariance is kept fixed. In Section 3.2, we show that this refinement applied jointly with prior precision refinement is in fact necessary to obtain a well-defined posterior predictive distribution when applying linearised Laplace to networks with normalisation layers.

Kristiadi et al. (2020) showed that reliable uncertainty estimates can be obtained by only applying the Laplace approximation to the last layer's weights. Daxberger et al. (2021b) avoided the need for KFAC approximations by only considering the curvature of a non-last-layer axis-aligned subspace of the weights. Although both of these methods are scalable to large models, they give a deterministic treatment to most network parameters, which eliminates the possibility of model selection.

Building upon the work of Khan et al. (2019), Immer et al. (2021a) demonstrated the application of the linearised Laplace model evidence for hyperparameter and architecture selection in neural networks. Their proposed method differs from the one studied in this paper however, in that (Immer et al., 2021a) apply an online approach; model hyperparameters are learnt simultaneously with network weights. The M step is performed every few epochs of NN training, using the unconverged neural network weights in the model evidence estimate. Although this objective is clearly biased, the use of a different Jacobian basis expansion at each M step brings the algorithm of Immer et al. (2021a) outside of the scope of the study of the present work. Be that as it may, the authors note that their method does not work well in the presence of normalisation layers. This motivates them to use FixUp regularisation (Zhang et al., 2019) instead. Our work finds that FixUp only partially

ameliorates the degradation in model selection capabilities brought by using the norm of the linearisation point in the model selection objective. Also related is the work of Maddox et al. (2020), who built upon the analysis of Mackay (1992) by showing that the Hessian eigenspectrum can be used to predict the generalisation properties of modern neural networks.

A.2. Approximate inference and uncertainty estimation with neural networks

As a more accurate alternative to the Laplace approximation, Neal (1995) introduced Hamiltonian Monte Carlo. Although asymptotically unbiased, this method requires a large number of full-batch gradient evaluations in order to draw independent samples. As a result, the cost of running Hamiltonian Monte Carlo in the modern large-network large-dataset setting is prohibitive. It is worth mentioning the work of Izmailov et al. (2021) who leverage vast computational resources to demonstrate the application of Hamiltonian Monte Carlo in the modern setting. Unlike the Laplace approximation, sampling methods do not provide straight forward access to marginal likelihood estimates. Neal (2001) introduced annealed importance sampling for this purpose, but its reliance on Hamiltonian Monte Carlo precludes its practical application to modern neural networks. Stochastic gradient sampling methods (Welling & Teh, 2011; Chen et al., 2014; Wenzel et al., 2020) relax the full batch requirement of the above methods but can be heavily biased as a result (Betancourt, 2015).

Another class of approximate inference methods for neural networks are those that employ variational inference (Hinton & van Camp, 1993). These re-formulate inference of the Bayesian posterior as the optimisation of the parameters of a variational distribution (Graves, 2011; Blundell et al., 2015; Osawa et al., 2019; Dusenberry et al., 2020). The objective being optimised is known as the Evidence Lower BOund (ELBO) and, in principle, it can be used as a surrogate objective for model selection. However, for neural network models, looseness in the constructed ELBOs has historically prevented these objectives from being used for this purpose. Two recent notable exceptions are the work on tighter variational bounds for NNs of Tomczak et al. (2021) and Ober & Aitchison (2021). However, their performance for model selection has yet to be evaluated extensively. Closely related to variational methods are those that use expectation propagation (Hernández-Lobato & Adams, 2015) and alpha divergence minimisation (Hernández-Lobato et al., 2016).

Finally, we note that although uncertainty estimation is the main goal of Bayesian deep learning, the most successful methods for uncertainty estimation with deep networks have traditionally been based on ensembling (Lakshminarayanan et al., 2017; Ashukha et al., 2020; Snoek et al., 2019; Dusenberry et al., 2020). The linearised Laplace method is unique in being one of the only Bayesian deep learning methods which performs competitively with ensembling methods (Daxberger et al., 2021b).

B. Example to illustrate definition 1

We now provide an example to illustrate how definition 1 constructs the parameter space as a direct sum of the subsapces of normalised and non-normalised weights.

Consider an MLP $f : \Theta \times \mathcal{X} \to \mathcal{Y}$ with a single input dimension $\mathcal{X} = \mathbb{R}$, single output dimension $\mathcal{Y} = \mathbb{R}$, single hidden layer and 2 hidden units. We apply layer norm after the input layer parameters. The model parameters are $\theta = [\theta_1, \theta_2, \theta_3, \theta_4] \in \Theta$ with θ_1, θ_2 belonging to the input layer and θ_3, θ_4 to the readout layer. We assume there are no biases without loss of generality.

Denoting the outputs of the first parameter layer $a = [\theta_1 x, \theta_2 x]$, layer norm applies the function

$$\frac{a - \mathbb{E}[a]}{\sqrt{\text{Var}(a)}}\gamma + \beta \quad \text{with} \quad \mathbb{E}[a] = 0.5\theta_1 x + 0.5\theta_2 x \quad \text{and} \quad \text{Var}(a) = 0.5(\theta_1 x - \mathbb{E}[a])^2 + 0.5(\theta_2 x - \mathbb{E}[a])^2$$

for $\gamma \in \mathbb{R}$, $\beta \in \mathbb{R}$. Now take $k \in \mathbb{R}_+$ to see that the output of the layernorm layer is invariant to scaling the input layer parameters by k

$$\frac{ka - \mathbb{E}[ka]}{\sqrt{\text{Var}(ka)}} = \frac{k(a - \mathbb{E}[a])}{k\sqrt{\text{Var}(a)}} = \frac{a - \mathbb{E}[a]}{\sqrt{\text{Var}(a)}}.$$

Thus, we have $f([\theta_1, \theta_2, \theta_3, \theta_4], \cdot) = f([k\theta_1, k\theta_2, \theta_3, \theta_4], \cdot)$. Now let Θ be the result of the internal *direct sum* $\Theta = \Theta' \oplus \Theta''$, and θ', θ'' be projections of θ onto the subspaces Θ' & Θ'', respectively, so that $\theta = \theta' + \theta''$. The operator $+$ is defined as the vector sum, as usual. In the simplest case where Θ' and Θ'' are aligned with the standard basis, this corresponds to vectors in Θ' having zero valued entries in the place of parameters to which normalisation is applied, $\theta' = [0, 0, \theta_3, \theta_4]$.

Vectors in Θ'' are non-zero for normalised parameters, $\theta'' = [\theta_1, \theta_2, 0, 0]$. Finally, we write the property of interest

$$f(\theta' + k\theta'', \cdot) = f(\theta' + \theta'', \cdot).$$

C. Additional proofs

We use notation defined in Section 4 of the main text. Additionally, we denote by $D_x f(y)$ the directional derivative of the function f in the direction of x, evaluated at the point y. This is the projection of the gradient of the function f onto the vector x, given by

$$D_x f(y) = \partial_z [f(z)](y) \cdot x. \tag{13}$$

C.1. Proof of lemma 4

Proof. First, we consider J'_k and J''_k. For J'_k, take any $\theta' \in \Theta'$ and consider the directional derivative $D_{\theta'} f(\tilde{\theta}' + k\tilde{\theta}'')$. From the limit definition,

$$\begin{aligned} D_{\theta'} f(\tilde{\theta}' + k\tilde{\theta}'') &= \lim_{\delta \downarrow 0} \frac{1}{\delta} \left[f(\tilde{\theta}' + \delta\theta' + k\tilde{\theta}''), \cdot) - f(\tilde{\theta}' + k\tilde{\theta}''), \cdot) \right] = \lim_{\delta \downarrow 0} \frac{1}{\delta} \left[f(\tilde{\theta}' + \delta\theta' + \tilde{\theta}''), \cdot) - f(\tilde{\theta}' + \tilde{\theta}''), \cdot) \right] \\ &= D_{\theta'} f(\tilde{\theta}' + \tilde{\theta}''). \end{aligned}$$

From the Jacobian-product definition, we have $J'_k \cdot \theta' = J' \cdot \theta'$. Since $\theta' \in \Theta'$ was arbitrary and we are working on a finite-dimensional Euclidean space, this shows $J'_k = J'$. For J''_k, consider $D_{\theta''} f(\tilde{\theta}' + k\tilde{\theta}'')$ for $\theta'' \in \Theta''$ arbitrary. We have

$$\begin{aligned} D_{\theta''} f(\tilde{\theta}' + k\tilde{\theta}'') &= \lim_{\delta \downarrow 0} \frac{1}{\delta} \left[f(\tilde{\theta}' + k\tilde{\theta}'' + \delta\theta''), \cdot) - f(\tilde{\theta}' + k\tilde{\theta}''), \cdot) \right] = \lim_{\delta \downarrow 0} \frac{1}{\delta} \left[f(\tilde{\theta}' + \tilde{\theta}'' + \frac{\delta}{k}\theta''), \cdot) - f(\tilde{\theta}' + \tilde{\theta}''), \cdot) \right] \\ &= \frac{1}{k} \lim_{\delta' \downarrow 0} \frac{1}{\delta'} \left[f(\tilde{\theta}' + \tilde{\theta}'' + \delta'\theta''), \cdot) - f(\tilde{\theta}' + \tilde{\theta}''), \cdot) \right] = \frac{1}{k} D_{\theta''} f(\tilde{\theta}' + \tilde{\theta}''). \end{aligned}$$

Repeating the same argument as for J'_k, we obtain $J''_k = \frac{1}{k} J''$.

Now, we look at the scaling of h_k. By definition, using that f is normalised and the previously derived scaling for J_k,

$$\begin{aligned} h_k(\theta' + k\theta'', \cdot) &= f(\tilde{\theta}' + k\tilde{\theta}'', \cdot) + J'_k(\theta' - \tilde{\theta}') + J''_k(k\theta'' - k\tilde{\theta}'') \\ &= f(\tilde{\theta}' + \tilde{\theta}'', \cdot) + J'(\theta' - \tilde{\theta}') + J''(\theta'' - \tilde{\theta}'') \\ &= h(\theta' + \theta'', \cdot), \end{aligned}$$

which is the claimed result.

For H_k, we examine it entry-wise. We have,

$$\begin{aligned} [H_k]_{mn} &= \partial_{\theta_m} \partial_{\theta_n} [L(h_k(\theta, \cdot))](\theta_k) = \partial_{\theta_m} \left(\sum_i \partial_h \ell(h_k(\theta_k, x_i), y_i) \cdot \partial_{\theta_n} [h(\theta, x_i)](\theta_k) \right) \\ &= \sum_i \partial_{\theta_m} [h_k(\theta, x_i)](\theta_k) \cdot \partial_h^2 \ell(h_k(\theta_k, x_i), y_i) \cdot \partial_{\theta_n} [h_k(\theta, x_i)](\theta_k) \\ &\quad + \sum_j \partial_h \ell(h_k(\theta_k, x_j), y_j) \cdot \partial_{\theta_m} \partial_{\theta_n} [h_k(\theta, x_j)](\theta_k). \end{aligned}$$

Now since h_k is affine, it has no curvature and thus $\partial_{\theta_m} \partial_{\theta_n} h_k(\theta, x)$ is identically zero for all $x \in \mathcal{X}$. With that, the second term in the sum vanishes. For the first sum, consider the *middle term*, the curvature of the negative log-likelihood function, and use $h_k(\theta_k, \cdot) = h(\theta, \cdot)$ to see that it is invariant to k. Finally, note that $\partial_{\theta_m} h_k$ and $\partial_{\theta_n} h_k$ are entries of J_k and inherit scaling from therein. Specifically, if both θ_m and θ_n belong to Θ'', we obtain $1/k^2$ scaling; if just one belongs to Θ'', we get $1/k$ scaling, and otherwise we obtain constant scaling. This completes the result for H_k. □

C.2. Proof of equation (11)

Consider a normalised network f, the linearisation point $\tilde{\theta}' + \tilde{\theta}''$, and the directional derivative with respect to parameters $\tilde{\theta}''$ in the direction of $\tilde{\theta}''$, denoted $D_{\tilde{\theta}''} f(\tilde{\theta}' + \tilde{\theta}'')$. On one hand, this is just the projection of the partial derivative of f with respect to θ'' evaluated at $\tilde{\theta}''$ and projected onto $\tilde{\theta}$, and thus $D_{\tilde{\theta}''} f(\tilde{\theta}' + k\tilde{\theta}'') = \partial_{\theta''} f(\tilde{\theta}, \cdot) \cdot \tilde{\theta}$. On the other hand, from the limit definition of the directional derivative,

$$D_{\tilde{\theta}''} f(\tilde{\theta}' + k\tilde{\theta}'') = \lim_{\delta \downarrow 0} \frac{1}{\delta} \left[f(\tilde{\theta}' + \delta\tilde{\theta}'' + k\tilde{\theta}''), \cdot) - f(\tilde{\theta}' + k\tilde{\theta}''), \cdot) \right] = \lim_{\delta \downarrow 0} \frac{1}{\delta} \left[f(\tilde{\theta}' + (\delta + k)\,\tilde{\theta}''), \cdot) - f(\tilde{\theta}' + k\tilde{\theta}''), \cdot) \right] = 0,$$

and thus $\partial_{\theta''} f(\tilde{\theta}, \cdot) \cdot \tilde{\theta} = 0$.

The quantity appearing in equation (11) is $\partial_{\theta''_{d:}} \phi(\tilde{\theta}''_{d:}, \cdot) \cdot \tilde{\theta}''_{d:}$. We now observe that for a fully normalised network, each of the outputs of the penultimate layer $[\phi(\theta_{d:}, \cdot)]_i$, with the output dimension being indexed by i, is a normalised network (definition 1) in its own right. Hence, we can apply the same reasoning as above to see that $\partial_{\theta''_{d:}} [\phi(\tilde{\theta}''_{d:}, \cdot)]_i \cdot \tilde{\theta}''_{d:} = 0$.

D. Algorithm for optimisation of the linear model loss

We now discuss the optimisation of the loss for the predictor $h(v, \cdot) = J(\cdot) \cdot v$ where $v \in \Theta$ is the linear model's parameter vector. We note that the procedure for the Taylor expanded model $f(\tilde{\theta}, \cdot) + J(\cdot) \cdot (v - \tilde{\theta})$ is analogous. We denote NN Jacobians as $J(\cdot) = \partial_\theta f(\tilde{\theta}, \cdot)$ and d_y is the output dimensionality $|\mathcal{Y}|$.

We wish to optimise v according to the objective $\mathcal{L}_{h,\Lambda}(v) = L(h(v, \cdot)) + \|v\|^2_\Lambda$. We adopt a first order gradient-based approach. We first consider the gradient of $L(h(v, \cdot)) = \sum_i \ell(J(x_i) \cdot v, y_i)$. Using the chain rule and evaluating at an arbitrary $\bar{v} \in \Theta$ we have

$$\partial_v [L(h(v, \cdot))](\bar{v}) = \sum_i \partial_v \ell(J(x_i) \cdot \bar{v}, y_i) \cdot \partial_v (J(x_i) \cdot \bar{v}) = \sum_i \partial_v \ell(J(x_i) \cdot \bar{v}, y_i) \cdot J(x_i).$$

Evaluating the affine function h consists of computing the Jacobian vector product $J(x_i)v$. This can be done while avoiding computing the Jacobian explicitly by using forward mode automatic differentiation or finite differences. We find both approaches to work similarly well, with finite differences being slightly faster, and forward mode automatic differentiation more numerically stable. Our experiments use finite differences, so we present this approach here. Specifically, we employ the method of Andrei (2009) to select the optimal step size. We then evaluate the loss gradient at the linear model output, denoting this vector in our algorithm as $g = \partial_v \ell(J(x_i) \cdot \bar{v}, y_i)$. This gradient can often be evaluated in closed form. Finally, we project g onto the weights by multiplying with the Jacobian. This vector Jacobian product is implemented using automatic differentiation. That is, $g^T J(x_i) = \partial_\theta [g^T \cdot f(\theta, x_i)](\tilde{\theta})$. We combine these steps in algorithm 1.

Evaluating the gradient of $\|v\|^2_\Lambda$ is trivial.

Algorithm 1: Efficient evaluation of the likelihood gradient for the linearised model

Inputs: Neural network f, Observation x, Linearisation point $\tilde{\theta}$, Weights to optimise v, Likelihood function $\ell(\cdot, y)$, Machine precision ϵ

1. $\delta = \sqrt{\epsilon}(1 + \|\tilde{\theta}\|_\infty)/\|v\|_\infty$ `// Set FD stepsize (Andrei, 2009)`
2. $\hat{y} = J(x) \cdot v \approx \frac{f(x, \tilde{\theta} + \delta v) - f(x, \tilde{\theta} - \delta v)}{2\delta}$ `// Two sided FD approximation to Jvp`
3. $g = \partial_v \ell(J(x) \cdot v, y)$ `// Evaluate gradient of loss at` $J(x) \cdot v$
4. $g^T \cdot J(x) = \partial_\theta [g^T \cdot f(\theta, x)](\tilde{\theta})$ `// Project gradient with backward mode AD`

Output: $g^T \cdot J(x)$

E. Additional discussion of recommendations and results

Here, we expand on the motivation behind the recommendations made in the main text and the implications of our results.

Motivation for evaluating the log-likelihood Hessian at $\tilde{\theta}$ instead of $\theta_\star$ In Section 3.1, we suggest evaluating the curvature of the linear model's loss function only once and around the linearisation point $\tilde{\theta}$. This yields $H = \partial^2_\theta [L(h(\theta, \cdot))](\tilde{\theta})$.

The more accurate alternative is to re-evaluate $\partial^2_\theta[L(h(\theta, \cdot))]$ at each optima of the linear model parameters found during the iterative procedure for computing $(\theta_\star, \Lambda_\star)$. This would require re-estimating the Hessian as many times as EM steps we perform.

However, we have good reason to believe that both approaches will yield similar results. The curvature of $L(h(\theta, \cdot))$ only depends on the linear model weights through the linear model's outputs, and the targets for likelihood functions outside of the exponential family. We assume our neural network model to be both flexible enough to fit the targets well and trained to do so. Furthermore, if we have a dense final layer, then the neural network's predictions $f(\tilde{\theta}, \cdot)$ are contained within the span of the linear model's basis expansion, as discussed in Section 3.3. Then, we can the expect the mode of the linear model's posterior to also fit the training data well and thus provide similar predictions to the neural network. If both models provide similar predictions, then the curvature of $L(h(\theta, \cdot))$ at $\tilde{\theta}$ and $\theta_\star$ will be similar.

Implications of the simplification of the linear model for linearised networks (12) Both the Taylor expanded affine model presented in equation (3) and the simplified linear model from equation (12) employ the same Jacobian basis expansion. However, their posterior modes $\theta_\star$, and thus optimal regularisation parameters $\Lambda_\star$, are different. Specifically, the constant-in-θ offset present in the Taylor expanded model pushes its posterior mean closer to the linearisation point $\tilde{\theta}$. As discussed in Section 3.3, scale invariant layers negate the effect of this offset. For fully normalised networks with a dense final layer, the offset is eliminated entirely and we recover the simplified linear model.

Accordingly, in our experiments, we find the simplified linear model's posterior mode to be further from the linearisation point than the Taylor expanded model's posterior mode. Thus, using the naïve linearised Laplace model evidence $\mathcal{M}_{\tilde{\theta}}$ results in poorer performance when using simplified linear model, i.e. when using normalisation layers. Even more interesting, however, is that employing the posterior mean of the simplified linear model in the model evidence estimate results in improved performance even when normalisation layers are not present, as shown in Section 5 and Appendix G.4.

On the use of FixUp instead of scale-invariant normalisation FixUp regularisation is used by Immer et al. (2021a) to avoid the need for normalisation layers in residual networks. FixUp specifies a network parameter initialisation scheme and adds a multiplicative scalar, and bias, at the output of each residual block. This multiplicative scalar introduces the following invariance; for a scaling of the weights in the residual block by $k > 0$ (applied as in Section 3.2) and a downscaling of the FixUp multiplicative scale factor by $1/k$, the neural network output remains unchanged. We may employ methods analogous to the ones presented in the main text to study this scenario. Although this study is beyond the scope of this work, we make two observations.

1. Unlike when employing normalisation layers, for a positive definite regulariser Λ, the invariance introduced by FixUp does not preclude the existence of a posterior mode nor does it result in the cancellation of terms in the affine Taylor expanded model (3) needed to recover the simplified linear model (12). As a result, we expect the posterior mode of the linearised model for FixUp networks to be closer to the linearisation point $\tilde{\theta}$ than for normalised networks. In turn, a naïve implementation of linearised Laplace will perform less poorly for FixUp networks than for normalised networks. Be that as it may, our results in Section 5 show there is still much to be gained from adopting recommendation 1.

2. Adept readers may note that a similar invariance is present in all neural networks with ReLU non-linearities; we can upscale the weights of a layer while downscaling the weights of any other layer and the neural network function is preserved. When using a positive definite regulariser Λ, this invariance does not present a large problem for the Laplace method in practise. FixUp is unique, however, in that the multiplicative parameter is one dimensional. Since the dimensionality of the parameters belonging to the preceding weight layer is likely to be much larger than one, the regularisation term in the neural network loss function $\mathcal{L}_{f,\Lambda}$ biases the parameters towards solutions where the multiplicative scaling factor is large while other network parameters take values close to 0. This bias can be understood as an implicit choice of scaling factor k, defined as in Section 3.2, with value depending on the size of the weight layers in residual blocks.

On batch norm Batch norm applies normalisation at the level of activations (Ioffe & Szegedy, 2015). That is, each activation to which batch norm is applied is normalised using statistics about said activation computed across a batch of data. This is distinct from other approaches to normalisation, such as group norm and layer norm, which compute statistics across network activations but separately for each individual observation. Consequently, batch norm introduces two peculiarities that are not shared by other normalisation layers.

1. When using batch norm, the parameters feeding into each activation are normalised separately. As a result, recommendation 2 prescribes regularising the weights feeding into each activation separately. However, empirically, we find that doing this provides no benefit over defining per-layer regularisation parameters. We hypothesize this is due to the scaling constant k (defined as in Section 3.2) being similar for weights corresponding to the same layers, perhaps due to the gradients reaching the different weights in each layer having similar variance (van Laarhoven, 2017). Thus, our experiments with batch norm implement layer-wise regularisation instead of activation-wise regularisation.

2. In the presence of batch normalisation, including network biases' gradients into the Jacobian feature expansion is not necessary in order to preserve the invariances described in Section 3.2. Because network biases represent a scalar offset to network activations, their effect is completely negated by batch normalisation. Indeed, for this reason, biases are most commonly omitted from weight layers followed by batch norm (He et al., 2016a).

On the use of weight decay and data augmentation for neural network training A common concern when performing probabilistic reasoning with neural networks is whether the regularisation techniques used for training break the formalisms put in place for the purpose of inference. For instance, data augmentation breaks any assumptions of observations being iid put in place for the speficication of a likelihood function (Nabarro et al., 2021). Conversely, some of the techniques required to satisfy the probabilistic inference formalism may degrade network training performance. This is the case for weight decay, which appears as the log of a zero-mean Gaussian prior over the network's parameters but can hinder learning in generative models.

The adaption of linearised Laplace presented in Section 3.1 of this paper provides a simple solution to the above qualms by decoupling neural network training from probabilistic inference. Applying recommendation 1 removes the need for the linearisation point $\tilde{\theta}$ to be a stationary point of the neural network posterior distribution. Thus, we are free to employ weight decay, a lack there-off, data augmentation, or any other regularisation schemes. In turn, properties introduced into our neural networks by training with these regularisation schemes will feature in the resulting linear model through the Jacobian basis expansion.

When performing inference in the linear model, we should take care to use a proper likelihood function and prior however. For this reason, we avoid data augmentation when computing the optima of the linear model loss $\theta_\star$ and when computing the linear model Hessian H.

F. Details on experimental setup

Here, we provide the details of our experimental setup which were omitted from the main text.

F.1. Experiments with full Hessian computation

This subsection concerns the experiments which use small architectures for which exact Hessian computation is tractable. These experiments are described in Section 5.1 and Section 5.2 of the main text. We first describe the setup components shared among architectures and then provide architecture-specific details. We exclude details for the U-net used in Section 5.2. Instead we provide these together with a brief description of the tomographic reconstruction task it performs in Appendix F.2.

Unless specified otherwise, NN weights $\tilde{\theta}$ are learnt using SGD, with an initial learning rate of 0.1, momentum of 0.9, and weight decay of 1×10^{-4}. We trained for 90 epochs, using a multi-step LR scheduler with a decay rate of 0.1 applied at epochs 40 and 70. This is a standard choice for CNNs and is default in the examples provided by Pytorch.

The linear weights $\theta_\star$ are optimised using Adam and with their gradients calculated using algorithm 1. We use a learning rate of 1×10^{-4} and train for 100 epochs. We set the initial regularisation parameter to be isotropic $\Lambda = \lambda I$ with $\lambda = 1 \times 10^{-4}$.

F.1.1. CNN

Our CNN is based on the LeNet architecture with a few variations found in more modern neural networks. The architecture contains 3 convolutional blocks, followed by global average pooling in the spatial dimensions, a flatten operation, and finally a fully-connected layer. The convolutional blocks consist of CONV $\rightarrow$ RELU $\rightarrow$ BATCHNORM. Instead of using max pooling layers, as in the original LeNet variants, we use convolutions with a stride of 2. The first convolution is 5×5, while the next two are 3×3. As described in the main text, we consider architectures of 3 different sizes. Table 3 shows the

number of the filters and number of parameters for each size of this model. The *Big* model's values where chosen to create a model as large as possible while keeping full-covariance Laplace inference tractable on one A100 GPU.

Table 3. Architecture parameters for the CNNs used in experiments.

	CONV1 FILTERS	CONV2 FILTERS	CONV3 FILTERS	PARAMS.	HESSAIN SIZE
Big	42	48	60	46 024	15.68 GB
Med	32	32	64	29 226	6.36 GB
Small	16	32	32	14 634	1.60 GB

F.1.2. RESNET, PRE-RESNET, AND BIASED-RESNET

Our ResNet is based on our CNN architecture. We replace the second and third convolutional blocks with residual blocks. The main branch of the residual blocks consist of CONV $\rightarrow$ BATCHNORM $\rightarrow$ RELU $\rightarrow$ CONV $\rightarrow$ BATCHNORM. We apply a final RELU layer after the residual is added. All of the convolutions in the residual blocks use the same number of filters. In order to downsample our features between blocks we use 1×1 convolutions with a stride of 2. Table 4 shows the number of the filters and number of parameters used for each size of this model.

Our Pre-ResNet architecture is identical to the ResNet except the main branch cosists of BATCHNORM $\rightarrow$ RELU $\rightarrow$ CONV $\rightarrow$ BATCHNORM $\rightarrow$ RELU $\rightarrow$ CONV, and we do not apply a RELU after adding the residual.

Table 4. Architecture parameters for the ResNets used in experiments.

	CONV1 FILTERS	RESBLOCK 1 FILTERS	RESBLOCK 2 FILTERS	PARAMS.	HESSAIN SIZE
Big	22	42	64	45 576	15.48 GB
Med	16	32	64	26 874	5.38 GB
Small	12	24	32	14 814	1.64 GB

Note that the standard ResNet architecture does not apply biases in the convolution layers. The only biases in the entire network are placed in the dense output layer. For our experiment where biases are included in the Jacobian feature expansion in Section 5.1, we modify the ResNet architecture to include biases in all convolutional layers in addition to the already present final dense layer bias. These biases account for a small increase in parameters, to 14 898, 26 986, and 45 726, in the small, medium, and big cases, respectively.

F.1.3. FIXUP RESNET

Our FixUp-ResNet architecture follows the standard ResNet structure described above, with the additional FixUP offsets and multipliers described in (Zhang et al., 2019). We also follow Zhang et al. (2019) in zero initializing the dense layer, and scaling the convolution weight initalization as a function of the depth of the network.

When training FixUp-ResNets, we use the Adam optimizer with a fixed learning rate of 0.01.

F.1.4. TRANSFORMER

Our Transformer architecture contains two encoder layers with two attention heads each, and no dropout. Its input is a sequence of tokens, to which we apply a linear embedding. We add a learnable `class` embedding for each input. This `class` token is used to classify the input. We do not use positional encoding, preserving permutation invariance in the input. The sizes of the embeddings and the MLP hidden dimensions are provided in Table 5.

When training Transformers, we use the Adam optimiser with a learning rate of 3×10^{-3}. We use an exponential learning rate decay with a gamma of 0.99 applied after every epoch of training.

F.2. U-Net tomographic reconstruction of KMNIST digits

In this section, we provide an overview of the tomographic reconstruction task for which we report results in Section 5.2. We also report the corresponding experimental details. Our setup almost exactly replicates that of Barbano et al. (2022a) and

Table 5. Architecture parameters for the Transformers used in experiments.

	MLP DIM	EMBEDDING DIM	PARAMS.	HESSAIN SIZE
Big	120	50	45 900	15.70 GB
Med	80	40	27 090	5.47 GB
Small	60	30	15 520	1.79 GB

Antorán et al. (2022), and we refer to these works for a fully detailed description.

Tomographic reconstruction aims to recover an unknown image $z \in \mathbb{R}^{d_z}$ from noisy measurements $y \in \mathcal{Y}$, which are often described by the application of a linear forward operator $\mathrm{A} \in \mathbb{R}^{d_y \times d_z}$ (in our case A is the discrete Radon Transform) and the addition of Gaussian noise $\eta \sim \mathcal{N}(0, \sigma_y^2 \, \mathrm{I}_{d_y})$ as

$$y = \mathrm{A}z + \eta, \tag{14}$$

and $|\mathcal{Y}| \ll d_z$. We replicate the setup of Barbano et al. (2022a); Antorán et al. (2022). That is, we use 10 test images from the KMNIST dataset, which consists of 28×28 gray-scale images of Hiragana characters (Clanuwat et al., 2018), we simulate y with 20 angles taken uniformly from the range $0°$ to $180°$, and add 5% white noise to $\mathrm{A}z$. We reconstruct z using the Deep Image Prior (DIP) (Ulyanov et al., 2018), which parametrises the reconstruction z as the output of a U-net $f(\theta)$ (Ronneberger et al., 2015). The reconstruction loss used to train the U-net reads

$$\|\mathrm{A}f(\theta) - y\|^2 + \zeta \mathrm{TV}(f(\theta)),$$

where the hyperparameter $\zeta > 0$ determines the strength of the Total Variation (TV) regularisation (Chambolle, 2004). This regulariser is used to train the U-net but then substituted for the Gaussian regulariser described in the main text for the purpose of inference in the linearised network. As discussed in Appendix E, this results in a valid inference scheme when recommendation 1 is applied.

We use the U-net like architecture deployed by Barbano et al. (2022a). Group norm is placed before every 3×3 convolution operation. The U-net architecture is a encoder-decoder, fully convolutional deep model constructing multi-level feature maps. We identify 3 distinct blocks for both the encoder branch and and 2 blocks for the the decoder branch: `In`, $\texttt{Down}_0$, $\texttt{Down}_1$, and $\texttt{Up}_0$ and $\texttt{Up}_1$, respectively. The `In` block consists of a 3×3 convolution. `Down` blocks consist of a 3×3 convolution with stride of 2 followed by a 3×3 convolution operation and a bi-linear up-sampling. The `Up` blocks instead consist of two successive 3×3 convolutional operations. Given the use of the leaky ReLU non-linearity, the normalised parameter groups of this network coincide with the described blocks. The number of channels is set to 32 at every scale. Multi-channel feature maps from the `In` block and from $\texttt{Down}_0$ are first transformed via a 1×1 convolutional operation to 4 channel feature maps and then fed to $\texttt{Up}_1$, $\texttt{Up}_0$. The reconstructed image is obtained as the output of $\texttt{Up}_1$ further processed via a 1×1 convolutional layer. The total number of parameters is 78k. This is too many for full Hessian construction on GPU but we get around this issue by performing inference in the lower dimensional space of observations, as described below. We refer to Antorán et al. (2022) for a full list of hyperparameters involved in training the U-net.

We construct a data-driven prior for the reconstruction z by adopting a linear model on the basis expansion given by the Jacobian of the trained U-net. However, differently from the procedure described in Section 2, we leverage the weight-space function-space duality of linear models (Rasmussen & Williams, 2006) to formulate our prior in the space of reconstructions directly

$$z = J\theta, \quad \theta \sim \mathcal{N}(0, \Lambda^{-1}) \quad \text{and thus} \quad z \sim \mathcal{N}(0, J\Lambda^{-1}J^\top).$$

This formulation allows us to perform (full-Hessian) Laplace inference at cost scaling cubically in $|\mathcal{Y}|$, which is much smaller that $|\Theta|$ for this task. We refer to Antorán et al. (2022) for details of the inference procedure employed.

The prior covariance Λ^{-1} is a filter-wise block-diagonal matrix which applies separate regularisation to the parameters of each block in the U-net, as described by Barbano et al. (2022a) and Antorán et al. (2022). For the single regulariser experiment, we keep the same prior structure but tie the marginal prior variance of all parameters. That is, we ensure all entries of the diagonal of Λ^{-1} are the same. The parameters of these regularisers are learnt via model evidence optimisation, as described in the main text.

F.3. Large scale experiments

For scaling linearised Laplace to large ResNets (i.e. ResNet-18 with 11M parameters and ResNet-50 with 23M parameters), we employ a Kronecker-factorisation of the Hessian/GGN. This is a common way to scale the Laplace approximation to large models (Daxberger et al., 2021a) and was originally proposed in Ritter et al. (2018). We use the recently-released `laplace` library[2] (Daxberger et al., 2021a) for fitting the KFAC Laplace models. For ResNets with batch norm, we use the reference implementation from the `torchvision` package[3]. For ResNets with fixUp, we use a popular open-source implementation[4]. To train the ResNet parameters, which will be used as the linearisation points $\tilde{\theta}$, we use the same hyperparameters as described at the top of Appendix F.1 for both batch norm and FixUp ResNets.

Finally, in Figure 6, we show that for the big ResNet model from Table 4 (i.e. the largest model for which we can tractably compute the full linear model Hessian), the optima of the model evidence induced by the KFAC approximation is similar to that obtained when using the full Hessian. This justifies the usage of KFAC Laplace for model evidence estimation with the full-scale ResNet models (where using the full Hessian is intractable).

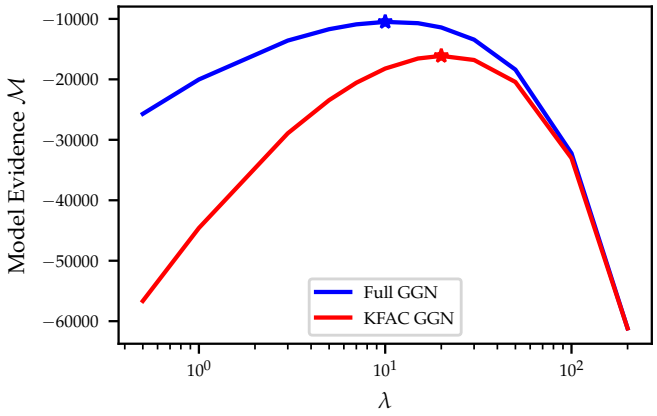

Figure 6. Linear model evidence for the big ResNet (cf. Table 4) plotted as a function of a single regularisation strength parameter λ, with $\Lambda = \lambda I$, for both the full Hessian and its KFAC approximation.

[2] https://github.com/AlexImmer/Laplace
[3] https://pytorch.org/vision/stable/models.html
[4] https://github.com/hongyi-zhang/FixUp

G. Additional experimental results

In this appendix we provide additional experimental results and analyses that are not included in the main text.

G.1. Hessian choice

Figure 7 shows the differences in the optimised layer-wise regularisation strength values when using the naïve objective $\mathcal{M}_{\tilde{\theta}}$ and the recommended objective $\mathcal{M}_{\theta_\star}$. We also compare using the Hessian of the linear model's loss evaluated at $\tilde{\theta}$ and at $\theta_\star$. The choice of hyperparameter optimisation objective has a very large impact on the resulting regularisation strength, with the naïve objective leading to much smaller regularisation strengths. As a result, the linear model's errorbars become too large and test performance deteriorates, as shown in Figure 4 in the main text. On the other hand, the choice of Hessian evaluation location has negligible impact on the learnt hyperparameters. Intuitively, the basis functions corresponding to weights closer to the network output are regularised less, since these features are more predictive of the targets. For this experiment we used the 46k parameter ResNet described in Appendix F.1.2 and the MNIST dataset.

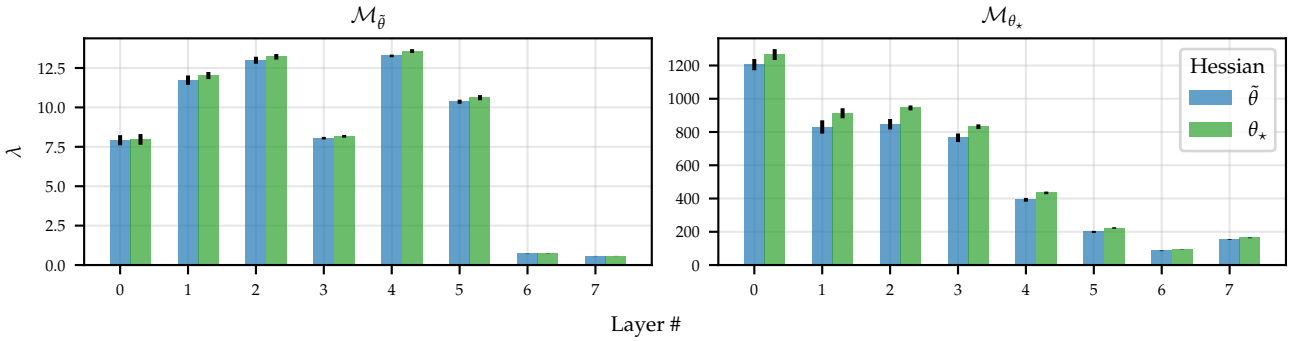

Figure 7. Layer-wise precision values learnt in the Hessian choice experiment. As before, we see that the choice of location around which the Hessian is calculated has a very small impact and the choice of parameters featuring in the norm term. The choice of naïve or recommended model evidence objective has a large impact. We note the much larger scale of the precisions in the case of the linear model weights.

G.2. Treatment of NN biases

Figure 8 shows the change in layer-wise regularisation strength values obtained when excluding the NN biases from the Jacobian basis expansion. The impact from the exclusion of biases is negligible while the application of recommendation 1 results in a difference of roughly two orders of magnitude in the obtained regularisation strengths.

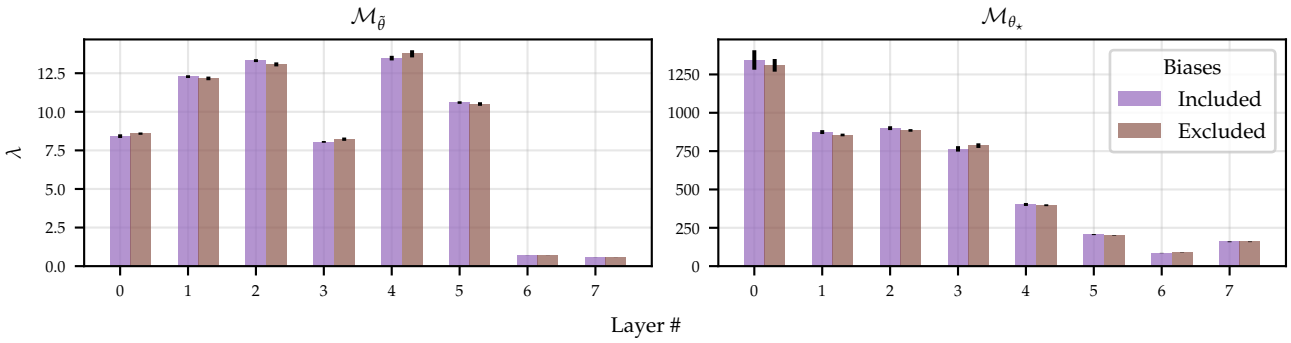

Figure 8. Layer-wise regularisation strengths learnt in the bias experiment of Figure 4. As before, we see that the choice of bias treatment has a smaller effect than the choice of weights featured in the norm of the model evidence. We also note the much larger scale of the precisions in the case of the linear model weights.

Figure 9 compares the empirical distributions of the NN linearisation point $\tilde{\theta}$ and linear model optima parameters $\theta_\star$. The parameters of the NN linearisation point present much larger variance than those of the linear model optima. Once again, we see that the exclusion of biases has little impact.

For this experiment we used the 46k parameter ResNet described in Appendix F.1.2 and the MNIST dataset.

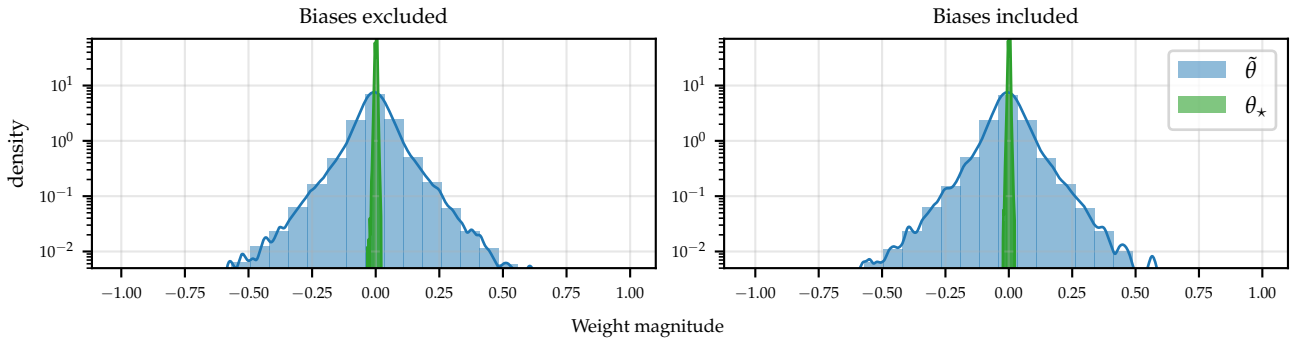

Figure 9. Histograms of the individual entries of $\tilde{\theta}$ and $\theta_\star$ for the models in the bias exclusion experiment of Figure 4.

G.3. Early stopping

Figure 10 shows the evolution of the layerwise optimised regularisation strength values for the Jacobian basis corresponding to the NN at different stages of training. We compare the regularisation parameters found with and without applying recommendation 1 and with and without batch norm. The results of Figure 5 are confirmed by the fact that that additional training of the NN does not cause the regulariser strengths found with $\mathcal{M}_{\theta_\star}$ and $\mathcal{M}_{\tilde{\theta}}$ to become closer to each other.

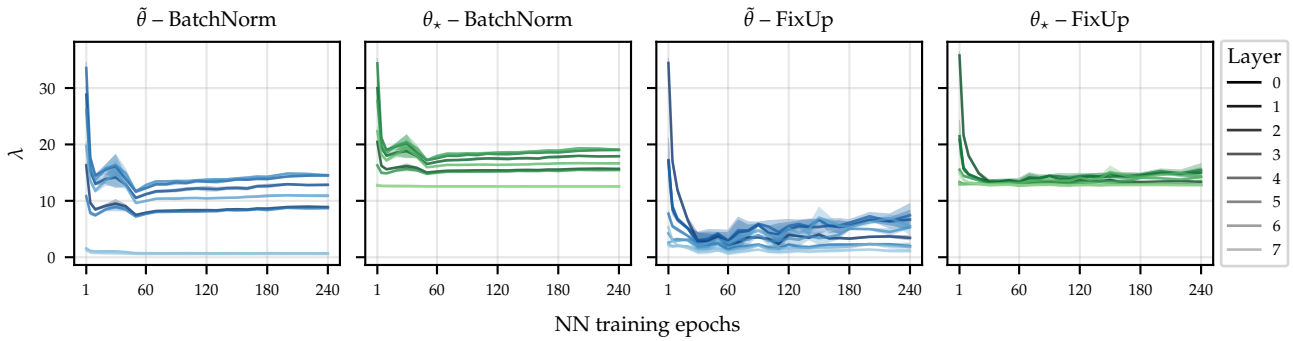

Figure 10. Evolution of precisions for the no early stopping experiment in Figure 5. In both the batch norm and fixUp cases, we see that as we train the NN for longer, the regularisation values obtained with $\mathcal{M}_{\theta_\star}$ and $\mathcal{M}_{\tilde{\theta}}$ converge to different values.

G.4. Validation of recommendations across architectures

Table 6 extends the results of Table 1 to two smaller sized models (*Med* and *Small*). We exclude the U-net architecture for which we only run a single model size with results already shown in Table 1. We also compare a number of schemes for obtaining the parameters that optimise of the linear model's loss.

Specifically, for each class of model, we compare the simplified linear model (i.e. (12)) optima $\theta_{*,\text{simple}}$ to the Taylor expanded model (i.e. (3)) optima $\theta_{*,\text{full}}$. We find that $\theta_{*,\text{simple}}$ performs equal to or (more often) better than $\theta_{*,\text{full}}$. For fully normalised networks, both approaches should be equivalent. However, the simplified model allows for more numerically stable optimisation of the weights. We can gain intuition for this by noting that fitting the Taylor expanded model's parameters can be viewed as regressing onto the neural network's residuals, which are likely to resemble noise. The more accurate parameter optima estimate obtained with the simplified linear model in turn yields a more accurate estimate of the model evidence objective.

For non-normalised FixUp models, the Taylor expanded model does not simplify, making the superior performance of the simplified objective intriguing. We conjecture that this is again a result of the simplified linear model being easier to optimise. We attempted to solve this problem by initialising the Taylor expanded model's weights at the linearisation point $\hat{\theta}$. We denote the results obtained with a model evidence featuring a set of parameters in its norm term found in this way

as $\theta_{*,\text{full,NN}}$. This tends to perform worse than initialising the Taylor expanded model's weights at 0, the default approach used in all other settings. This result tells us that the Taylor expanded model's parameter optima is likely closer to that of simplified linear model than to the NN linearisation point. However, we can not discard the optimisation instability hypothesis.

As before, recommendation 1 (using $\mathcal{M}_{\theta_*}$ rather than $\mathcal{M}_{\tilde{\theta}}$) yields notable improvements in most cases. Also in most cases, recommendation 2 (using per-layer prior precisions in normalised networks) yields moderate improvements or does not degrade performance for normalised networks.

Table 6. Extension of Table 1 to medium and small sized models, as well as to different methods for optimisation of the linear models' parameter optima.

			TRANSFORMER	CNN	RESNET	PRE-RESNET	FIXUP
Big	$\mathcal{M}_{\theta_\star,\text{simple}}$	single λ	0.162 $\pm$ 0.042	0.025 $\pm$ 0.000	0.017 $\pm$ 0.000	0.017 $\pm$ 0.000	0.020 $\pm$ 0.001
		multiple λs	0.162 $\pm$ 0.042	0.025 $\pm$ 0.000	0.016 $\pm$ 0.001	0.016 $\pm$ 0.000	0.018 $\pm$ 0.000
	$\mathcal{M}_{\theta_\star,\text{full}}$	single λ	0.161 $\pm$ 0.042	0.073 $\pm$ 0.001	0.083 $\pm$ 0.003	0.087 $\pm$ 0.001	0.055 $\pm$ 0.006
		multiple λs	0.162 $\pm$ 0.042	0.073 $\pm$ 0.001	0.066 $\pm$ 0.002	0.064 $\pm$ 0.002	0.061 $\pm$ 0.005
	$\mathcal{M}_{\theta_\star,\text{full,NN}}$	single λ	0.161 $\pm$ 0.042	0.211 $\pm$ 0.002	0.149 $\pm$ 0.004	0.117 $\pm$ 0.002	0.060 $\pm$ 0.006
		multiple λs	0.161 $\pm$ 0.042	0.169 $\pm$ 0.002	0.138 $\pm$ 0.004	0.109 $\pm$ 0.003	0.070 $\pm$ 0.011
	$\mathcal{M}_{\tilde{\theta}}$	single λ	0.310 $\pm$ 0.060	0.253 $\pm$ 0.001	0.252 $\pm$ 0.006	0.220 $\pm$ 0.004	0.153 $\pm$ 0.021
		multiple λs	0.162 $\pm$ 0.042	0.205 $\pm$ 0.002	0.236 $\pm$ 0.005	0.239 $\pm$ 0.004	0.200 $\pm$ 0.018
Med	$\mathcal{M}_{\theta_\star,\text{simple}}$	single λ	0.132 $\pm$ 0.026	0.024 $\pm$ 0.000	0.017 $\pm$ 0.000	0.018 $\pm$ 0.000	0.017 $\pm$ 0.000
		multiple λs	0.133 $\pm$ 0.026	0.023 $\pm$ 0.000	0.016 $\pm$ 0.000	0.017 $\pm$ 0.000	0.038 $\pm$ 0.021
	$\mathcal{M}_{\theta_\star,\text{full}}$	single λ	0.132 $\pm$ 0.025	0.060 $\pm$ 0.001	0.059 $\pm$ 0.002	0.055 $\pm$ 0.002	0.062 $\pm$ 0.003
		multiple λs	0.132 $\pm$ 0.026	0.059 $\pm$ 0.001	0.050 $\pm$ 0.001	0.047 $\pm$ 0.000	0.079 $\pm$ 0.024
	$\mathcal{M}_{\theta_\star,\text{full,NN}}$	single λ	0.132 $\pm$ 0.025	0.154 $\pm$ 0.001	0.097 $\pm$ 0.002	0.074 $\pm$ 0.002	0.072 $\pm$ 0.005
		multiple λs	0.132 $\pm$ 0.025	0.124 $\pm$ 0.002	0.087 $\pm$ 0.001	0.072 $\pm$ 0.002	0.065 $\pm$ 0.003
	$\mathcal{M}_{\tilde{\theta}}$	single λ	0.241 $\pm$ 0.028	0.190 $\pm$ 0.002	0.165 $\pm$ 0.005	0.132 $\pm$ 0.004	0.183 $\pm$ 0.008
		multiple λs	0.133 $\pm$ 0.025	0.153 $\pm$ 0.002	0.176 $\pm$ 0.004	0.167 $\pm$ 0.001	0.402 $\pm$ 0.213
Small	$\mathcal{M}_{\theta_\star,\text{simple}}$	single λ	0.112 $\pm$ 0.016	0.025 $\pm$ 0.000	0.017 $\pm$ 0.001	0.018 $\pm$ 0.001	0.018 $\pm$ 0.000
		multiple λs	0.112 $\pm$ 0.016	0.024 $\pm$ 0.000	0.018 $\pm$ 0.000	0.018 $\pm$ 0.000	0.016 $\pm$ 0.001
	$\mathcal{M}_{\theta_\star,\text{full}}$	single λ	0.112 $\pm$ 0.016	0.045 $\pm$ 0.001	0.040 $\pm$ 0.001	0.040 $\pm$ 0.001	0.055 $\pm$ 0.003
		multiple λs	0.112 $\pm$ 0.016	0.045 $\pm$ 0.000	0.038 $\pm$ 0.000	0.035 $\pm$ 0.001	0.051 $\pm$ 0.001
	$\mathcal{M}_{\theta_\star,\text{full,NN}}$	single λ	0.112 $\pm$ 0.016	0.085 $\pm$ 0.001	0.059 $\pm$ 0.001	0.048 $\pm$ 0.001	0.060 $\pm$ 0.003
		multiple λs	0.112 $\pm$ 0.016	0.075 $\pm$ 0.001	0.054 $\pm$ 0.001	0.047 $\pm$ 0.001	0.060 $\pm$ 0.005
	$\mathcal{M}_{\tilde{\theta}}$	single λ	0.208 $\pm$ 0.021	0.108 $\pm$ 0.001	0.095 $\pm$ 0.002	0.085 $\pm$ 0.002	0.161 $\pm$ 0.011
		multiple λs	0.114 $\pm$ 0.016	0.095 $\pm$ 0.001	0.118 $\pm$ 0.002	0.119 $\pm$ 0.003	0.178 $\pm$ 0.003

G.5. U-Net tomographic reconstruction

We illustrate the tomographic reconstruction task in Figure 11 by showing the results obtained for one of the KMNIST images used in our experiments. For this example, we use layerwise regularisation parameters, following recommendation 2, but ablate recommendation 1. That is, we compare the use of $\mathcal{M}_{\tilde{\theta}}$ and $\mathcal{M}_{\theta_\star}$. The former approach not only produces much larger pixel-wise uncertainties but also assigns uncertainty to large portions on the reconstructed image. Applying recommendation 1 yields smaller magnitude errorbars with the placement high uncertainty pixels matching the regions of increased reconstruction error much more closely. The histogram reveals the latter approach to result in much better calibrated uncertainty estimates.

We further investigate the difference between the two objectives $\mathcal{M}_{\tilde{\theta}}$ and $\mathcal{M}_{\theta_\star}$ in Figure 12 by plotting the evolution of the block-wise regularisation strengths during successive iterations of these parameters' optimisation. The objective $\mathcal{M}_{\tilde{\theta}}$ results

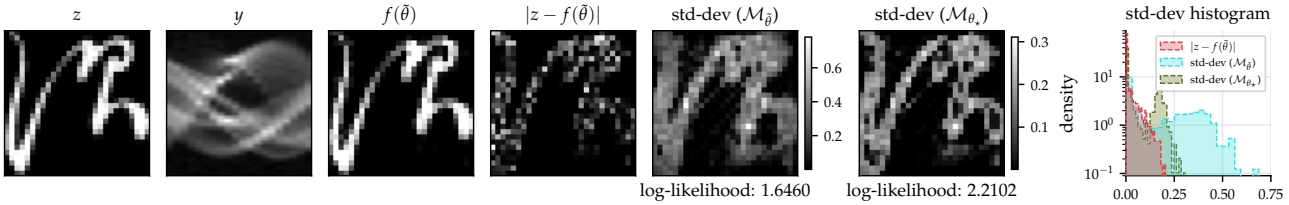

Figure 11. Enumerating the 7 plots from left to right: 1. Original KMNIST image. 2. Low dimensional projection obtained from applying (14). 3. Tomographic reconstruction from our U-net. 4. Associated pixel error in absolute value. 5. The pixel-wise marginal predictive standard deviation obtained when employing $\mathcal{M}_{\tilde{\theta}}$. 6. Same for $\mathcal{M}_{\theta_\star}$. 7. Histograms comparing reconstruction error with predictive standard deviation for both approaches under consideration.

in most blocks' regularisation parameter λ going to 0 (and thus the marginal prior variances diverging) for all 10 test images used in our experiments. This is due to the norm of the linearisation point $\tilde{\theta}$ being very large for the U-net. On the other hand, the $\mathcal{M}_{\theta_\star}$ objective is well behaved.

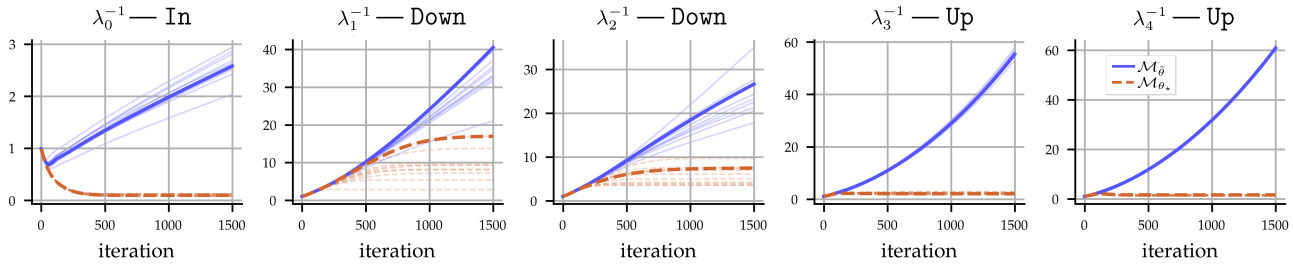

Figure 12. Hyperparameter optimisation traces for our U-net. Each figure corresponds to the regularisation parameter for a different convolutional block described in Appendix F.2. The thick lines correspond to optimisation for the image shown in Figure 11. Thin transparent lines correspond to the other 9 images used in our experiments. $\mathcal{M}_{\tilde{\theta}}$ leads hyperparameters to diverge, while $\mathcal{M}_{\theta_\star}$ converges.

G.6. Image classification with full-scale ResNets

In Figure 13 and Figure 14, we plot the model evidence as well as test log-likelihoods (minus the NLL reported in Section 5) obtained by the different models as a function of a single regularisation parameter λ, with $\Lambda = \lambda I$. Figure 13 considers ResNet-18 trained on MNIST, while Figure 14 considers ResNet-50 trained on CIFAR10. For both ResNets, we consider batch norm and FixUp . We can see that in both settings, the optima of $\mathcal{M}_{\theta_\star}$ corresponds to a larger λ than that of $\mathcal{M}_{\tilde{\theta}}$. The former provides a higher test log-likelihood. For the FixUp models, we further consider both the simplified and the full Taylor expanded linearised models (denoted $\mathcal{M}^\star_{\text{simple}}$ and $\mathcal{M}^\star_{\text{full}}$ respectively) for the purpose of finding the posterior mode $\theta_\star$ (recall that both linear models are equivalent for normalised networks). We see that the simple linear model finds larger λ values than the Taylor expanded model and obtains better test log-likelihood. Finally, Table 7 is analogous to Table 2 in that it compares the test log-likelihoods when using a single λ value versus using a separate λ values per layer, but reports results for ResNet-18 on MNIST. As in Table 2, we observe that using multiple λ values generally yields better results.

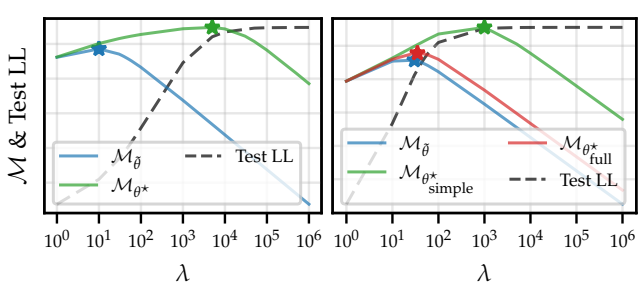

Figure 13. ResNet-18 with BatchNorm (left) and FixUp (right) on MNIST. Model evidence and test log-likelihood as a function of λ.

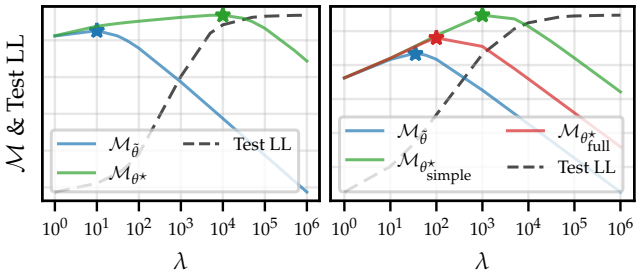

Figure 14. ResNet-50 with BatchNorm (left) and FixUp (right) on CIFAR10. Model evidence and test log-likelihood as a function of λ.

Table 7. Test negative log-likelihoods for ResNet-18 on MNIST.

		BatchNorm	FixUp
$\mathcal{M}_{\theta_{\star,\text{simple}}}$	single λ	**0.003** ± 0.000	0.007 ± 0.001
	multiple λs	**0.003** ± 0.000	**<u>0.004</u>** ± 0.000
$\mathcal{M}_{\theta_{\star,\text{full}}}$	single λ	0.166 ± 0.019	0.143 ± 0.013
	multiple λs	<u>0.100</u> ± 0.015	<u>0.100</u> ± 0.012
$\mathcal{M}_{\tilde{\theta}}$	single λ	0.529 ± 0.009	0.288 ± 0.015
	multiple λs	0.520 ± 0.009	<u>0.237</u> ± 0.017