4th Symposium on Advances in Approximate Bayesian Inference, 2022 1–18

Linearised Laplace Inference in Networks with Normalisation Layers and the Neural g-Prior

Javier Antorán JA666@CAM.AC.UK
University of Cambridge

James Urquhart Allingham JUA23@CAM.AC.UK
University of Cambridge

David Janz DJANZ@UALBERTA.CA
University of Alberta

Erik Daxberger EAD54@CAM.AC.UK
University of Cambridge
Max Planck Institute for Intelligent Systems

Eric Nalisnick E.T.NALISNICK@UVA.NL
University of Amsterdam

José Miguel Hernández-Lobato JMH233@CAM.AC.UK
University of Cambridge

Abstract

We show that for neural networks (NN) with normalisation layers, i.e. batch norm, layer norm, or group norm, the Laplace model evidence does not approximate the volume of a posterior mode and is thus unsuitable for model selection. We instead propose to use the Laplace evidence of the linearized network, which is robust to the presence of these layers. We also identify heterogeneity in the scale of Jacobian entries corresponding to different weights. We ameliorate this issue by extending the scale-invariant g-prior to NNs. We demonstrate these methods on toy regression, and image classification with a CNN.

1. Introduction

Normalisation layers, for example batch norm (Ioffe and Szegedy, 2015) or layer norm (Ba et al., 2016), are ubiquitous in modern NN architectures (He et al., 2016; Vaswani et al., 2017). They speed up training and make it robust to the choice of hyperparameters. They also introduce an invariance to the scale of the weights.

We study the effects of this invariance on probabilistic inference, in particular in the context of the linearised Laplace approximation, introduced by Mackay (1992). This method approximates a NN with a surrogate linear model where the NN's Jacobian acts as a feature expansion (Khan et al., 2019). The linearised model shares its predictive mean with the NN. Linearised Laplace predictive uncertainty is simple to compute (Immer et al., 2021b) and performs strongly on uncertainty quantification tasks (Kristiadi et al., 2020; Daxberger et al., 2021b,a). The Laplace model evidence or marginal log-likelihood (MLL) can also be computed in closed form (Mackay, 1992) providing an objective for model selection.

Daxberger et al. (2021b) find that the quality of uncertainty estimates provided by linearised Laplace is heavily dependent on the choice of Gaussian prior precision. They

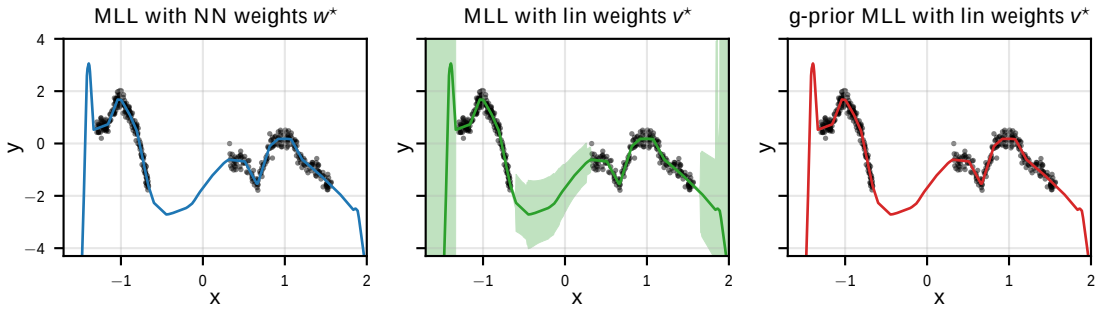

Figure 1: Fit for a 2 hidden-layer MLP with layer norm and residual connections. For the left and middle plots, a prior precision $\Lambda{=}\lambda I$ is set through MLL optimisation. **Left:** Using the MLL in eq. (3) leads to too large errorbars. **Middle:** Using the proposed MLL eq. (9) fixes the issue. **Right:** The g-prior (g optimised with eq. (9)) results in smoother errorbars.

select an isotropic precision using cross validation. The MLL is a preferable alternative, as it does not require a held out set and scales to more hyperparameters (Immer et al., 2021a).

We show that in normalised networks, the point around which the NN is linearised can not be a mode of the posterior for neither the NN model nor the linearised NN. This results in the traditional formulation of the Laplace MLL returning spurious values, rendering it unsuitable for model selection. We show how to find the mode of the linearised NN posterior and use it to propose a linearised Laplace MLL objective that is robust to the presence of normalisation layers. Next, we observe heterogeneity in the scales of the NN Jacobian entries corresponding to different weights. In this setting, isotropic priors can be excessively restrictive. We address this by extending the scale invariant g-prior (Zellner, 1996) to the NN setting. This prior allows us to obtain more expressive predictive posteriors, shown in Figure 1. We validate our proposals, first on a 1D toy problem, and then on an image classification task, where we evaluate in and out of distribution (OOD) performance.

2. Preliminaries

We consider a supervised learning setting with observations $(x_i, y_i)_{i=1}^n$, $x_i \quad \mathrm{R}^{d_x}$ and y_i R^{d_y}. We estimate y_i from x_i using a NN $f(x_i, w)$ with parameters a column vector w R^{d_w}. To do so, we maximise an objective of the form $G_f(w) + R(w)$, where $G_f(w) = \sum_{i \le n} g(y_i, f(x_i, w))$ is a data fit term and $R(w)$ is a regulariser that encourages the entries of w to be small. We denote by w the output of some stochastic gradient optimisation algorithm (SGD) applied to this objective. Within the Bayesian framework, g and R are obtained by assuming a probabilistic model for which $g(y_i, \hat{y}_i)$ is the log-likelihood $\log p(y_i \mid \hat{y}_i)$ and $R(w)$ is the log prior $\log \pi(w)$. Thus, w is usually understood to be a *maximum a posteriori* (MAP) solution or mode of the resulting posterior $p(w \mid (x_i, y_i)_{i=1}^n)$.

Linearised Laplace Inference for NNs The linearised Laplace method for approximating the intractable posterior is based on a 1^{st} order Taylor expansion of f around w

$$f(x, w\ ) + \quad {}_w f(x, w\ )(v - w\ ) =: h(x, v), \tag{1}$$

where $J(x) =: \quad {}_w f(x, w)|_{w=w} \quad \mathrm{R}^{d_y \times d_w}$ is the Jacobian of f with respect to the parameters. The prior over $v \quad \mathrm{R}^{d_w}$ is chosen to match that of the NN weights. The posterior is then modelled as a Gaussian with mean $\mu = w$ and covariance matrix $\Sigma = (H + \Lambda)^{-1}$

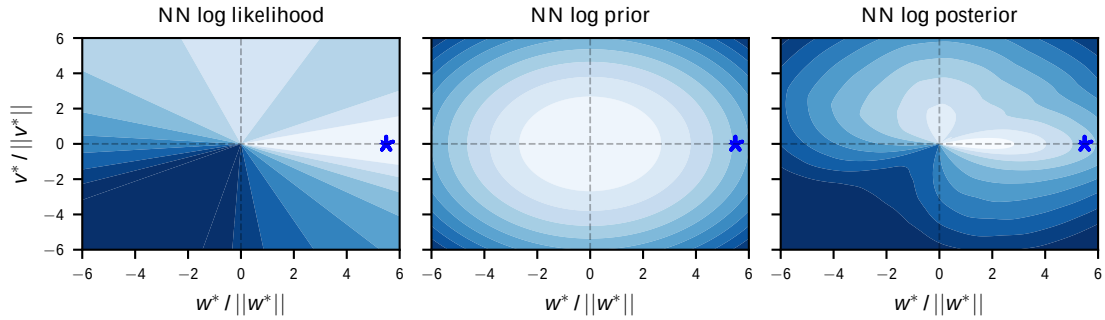

Figure 2: Log likelihood $G_f(w)$, log prior $\log \pi(w)$ and posterior $G_f(w) + \log \pi(w)$ landscape projection for the input layer of the NN used in Figure 1. F corresponds to the weight setting found with SGD, w . The horizontal projection axis corresponds to the direction of w while the vertical to some other vector v , both orthonormalised.

where $\Lambda = - \frac{2}{w} \log \pi(w)|_{w=w}$ $\mathbb{R}^{d_w \times d_w}$ is the Hessian (or precision) of the log prior and

$$H = \prod_{i \le n} J(x_i)^T H(x_i, y_i) J(x_i) \quad \text{where} \quad H(x, y) = - \frac{2}{\hat{y}} g(y, \hat{y})|_{\hat{y}=f(x,w)}, \tag{2}$$

the Hessian of the log-likelihood. We assume Λ is positive definite and let H be positive semi-definite. Under linearised Laplace, $\mathbb{E}[h(x, v)] = f(x, w)$ and $\mathrm{Var}[h(x, v)] = J(x) \frac{2}{\Sigma}$.[1]

Laplace Model Evidence Approximation We adopt the common Gaussian prior $N(0, \Lambda^{-1})$. The precision Λ is usually chosen to maximise the volume of a Gaussian approximation to the posterior mode containing w . This so-called *model evidence* is given by

$$\frac{1}{2} \Big[- w \frac{2}{\Lambda} - \log \Big[\frac{\det(H + \Lambda)}{\det(\Lambda)} \Big] \Big] + C, \tag{3}$$

where C is independent of Λ, and w is the linearisation point obtained with an optimisation algorithm. The norm term encourages the prior precision Λ to be small enough to accommodate w . The determinant ratio encourages larger precisions such that the contraction of the posterior's variance relative to the prior's is minimised.

3. Pathologies in Linearised Laplace with Normalised Networks

We suppose there exists a partition of the neural network parameters w into $L + 1$ groups $w_1, \ldots, w_{L+1}$, e.g. layers, and that the output of the network is invariant to the scaling of groups of parameters $w_1, \ldots, w_L$. That is, taking $\mathbb{R}_+ = \{x \quad \mathbb{R} : x > 0\}$, for all $k \quad \mathbb{R}_+^L$,

$$f(x, w) = f(x, w_{L+1} \quad k \cdot w_{1:L}), \tag{4}$$

where $k \cdot w_{1:L} = \prod_{i=1}^{L} k_i w_i$ refers to set-element-wise multiplication in the rest of the paper. Figure 2 illustrates this radial invariance. We call a neural network satisfying (4) a *normalised neural network*. This invariance occurs in, for example, layer norm (Ba et al., 2016), group norm (Wu and He, 2020), batch norm (Ioffe and Szegedy, 2015), or even in so-called normalisation-free methods (Brock et al., 2021). Group w_{L+1}, which corresponds to the output layer of the neural network, is not invariant.

1. For B a positive semidefinite matrix, $\|a\|_B$ denotes the norm given by $a^T Ba$.

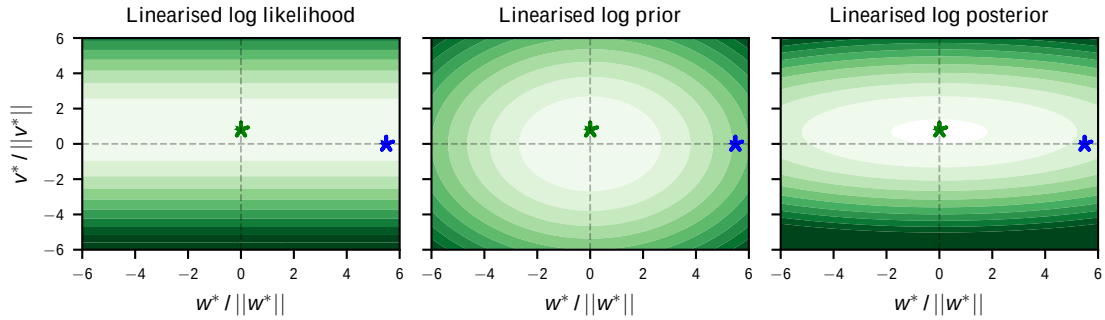

Figure 3: Log likelihood $G_h(v)$, log prior $\log \pi(v)$ and posterior $G_h(v) + \log \pi(v)$ for the input layer weights of the *linearised* NN from Figure 1. F corresponds to w and F to v. The projection is onto these two directions. The posterior mode v is different from w.

Recall that $G_f(w) = \sum_{i \leq n} \log p(y_i | f(x_i, w))$ and $\pi(w) = \mathcal{N}(w; 0, \Lambda^{-1})$. For f, a normalised neural network, a maximum of $G_f(w) + \log \pi(w)$, also known as a *maximum a posteriori, does not exist*. To see this, suppose w is a MAP. Take $w = w_{L+1} \quad (0.5 \cdot w_{1:L})$. Then

$$G_f(w) = G_f(w) \quad \text{and} \quad \log \pi(w) > \log \pi(w) \tag{5}$$

and thus w is not a MAP solution. Additionally, let w_l be a normalisation group of w (as used in eq. (4)), with $l \leq L$, and corresponding Jacobian entries $J_l(x) = \nabla_{w_l} f(x, w)|_{w=w}$. *Then w_l lies in the null space of $J_l(x)$.* This can be seen by taking the directional derivative

$$J_l(x), w_l = \lim_{\delta \to 0} \frac{1}{\delta} \sum f(x, w_{\setminus l} \quad ((1+\delta) \cdot w_l)) - f(x, w) \overset{\text{eq. (4)}}{=} 0. \tag{6}$$

Similar results have been noted in the optimisation literature, where the prior term is introduced through weight decay (van Laarhoven, 2017; Hoffer et al., 2018; Cai et al., 2019; Li et al., 2020; Lobacheva et al., 2021). As a consequence of eq. (6), the gradient of G_f acts like a tangential force, making the norm of w larger. The prior gradient acts like a centripetal force, keeping the norm small. This can be seen in Figure 2. van Laarhoven (2017) shows that SGD leads to a norm at convergence of $\|w\|_2^2 \propto (\sigma(\nabla_w G_f)/\lambda)^{0.5}$, where $\sigma(\nabla_w G_f)$ is the standard deviation of $\nabla_w G_f$ and λ is the precision of an isotropic Gaussian prior. This can be understood as SGD returning $w_{L+1} \quad k \cdot w_{1:L}$, where $k \quad \mathbb{R}_+^L$ depends on extraneous factors such as the learning rate or batchsize.

How does the above pertain to linearised Laplace? Denote by $G_h(v) = \sum_{i \leq n} \log p(y_i | h(x_i, v))$ the log likelihood function of the linearised model eq. (1), and recall that $\pi(v) = \mathcal{N}(v; 0, \Lambda^{-1})$. *The linearisation point w is not the maximum of $G_h(v) + \log \pi(v)$.* To see this, suppose that w is the MAP and take w to be $w_{L+1} \quad (0 \cdot w_{1:L})$. We have

$$\begin{aligned} h(x, w) &= f(x, w) + J(x)((w_{L+1} \quad 0) - w) = f(x, w) - J_{1:L}(x) w_{1:L} \overset{\text{eq. (6)}}{=} f(x, w) \\ &= h(x, w) \quad \text{thus} \quad G_h(w) = G_h(w) \quad \text{and} \quad \log \pi(w) > \log \pi(w), \end{aligned} \tag{7}$$

and w is not a MAP solution for the linearised model. The Laplace method relies on w being a posterior mode such that the first order term in a quadratic expansion of the posterior vanishes. As a consequence of eqs. (5) and (7), the *Laplace evidence from eq. (3) does not approximate the volume of a posterior mode for neither the NN nor linearised NN*, making it unsuitable for model selection. In practice, maximising eq. (3) leads to arbitrary results due to the dependence on the norm of the SGD solution, as we will see in Section 6.

Why do Daxberger et al. (2021b) see such strong empirical performance with linearised Laplace in normalised NN architectures despite the above? The key insight is that *the predictive variance* $\mathrm{Var}[h(x, v)] = \quad J(x) \; {}^{2}_{\Sigma}$, *does not depend directly on the norm of* w . Because Daxberger et al. (2021b) employ cross-validation, as opposed to the model evidence (MLL), to select Λ, they do not encounter the issues described above.

In summary, SGD provides us with a solution w of arbitrary norm which may be a mode of the likelihood but is *not* a mode of the posterior (MAP solution) for either the normalised NN or its corresponding (tangent) linear model. Given a suitable prior precision, this does not create issues when estimating uncertainty because the posterior predictive variance is invariant to the norm of the linearisation point w , only depending on its angle. However, normalisation breaks the Laplace MLL objective, which relies on the norm of w as a measure of model complexity.

4. Finding a MAP to the Lost Linearised Evidence

We proceed to obtain the evidence for the linearised model corresponding to a normalised network. Its posterior landscape is shown in Figure 3. First, we note that *for a NN normalised as in eq. (4), and with a fully connected output layer, the first order expansion matches a simple basis function linear model*

$$h(x, v) = f(x, w\ ) + J(x)(v - w\ ) \overset{\text{eq. (6)}}{=} f(x, w\ ) + J(x)v - J_{L+1}(x)w_{L+1} = J(x)v. \quad (8)$$

The final equality is due to the Jacobian entries corresponding to the last layer weights matching the last layer activations: $f(x, w\ ) = J_{L+1}(x)w_{L+1}$. Referring to the MAP solution of $G_h(w) + \pi(w)$ as v , we write the linear model's Laplace MLL objective as

$$\log \int \prod_{i \le n} \mathcal{N}(y_i; h(x_i, v), H(x_i, y_i)^{-1}) d\pi = \frac{1}{2} \left(- \ v \ {}^{2}_{\Lambda} - \log \frac{\det(H + \Lambda)}{\det(\Lambda)} \right) + C. \quad (9)$$

By eq. (7) we know that $v \ = w$. From eq. (8) we see that v corresponds to the MAP slope of a regression hyperplane on the Jacobian basis. Intuitively, the prior density $\pi(v\ )$ can be interpreted as a measure of model complexity, unlike $\pi(w\ )$.

The large size of our NN's weight space or use of non-linear linking functions often precludes finding v in closed form. Instead, we propose algorithm 1 for evaluating the gradient $_v \log p(y|h(x, v))$ without explicitly computing Jacobians. We derive it in Appendix B.

Algorithm 1: Efficient gradient evaluation for linearised model

Inputs: Neural network f, Observation x, Linearisation point w , Weights to optimise v, Likelihood function $p(y|\cdot)$, Machine precision ε

$\delta = \sqrt{\varepsilon}(1 + \ w \ _\infty)/ \ v \ _\infty$ // Set FD stepsize (Andrei, 2009)

$\hat{y} = J(x)v \approx \frac{f(x, w\ + \delta v) - f(x, w\ - \delta v)}{2\delta}$ // Two sided FD approximation to Jvp

$g_y = \ _{\hat{y}} \log p(y|\hat{y})|_{\hat{y}=J(x)v}$ // Evaluate gradient of loss at $\hat{y} = J(x)v$

$g_v = g_y^T J(x) = \ _w (g_y^T f(x, w))|_{w=w}$ // Project gradient with backward mode AD

Output: $_v \log p(y|h(x, v)) = g_v$

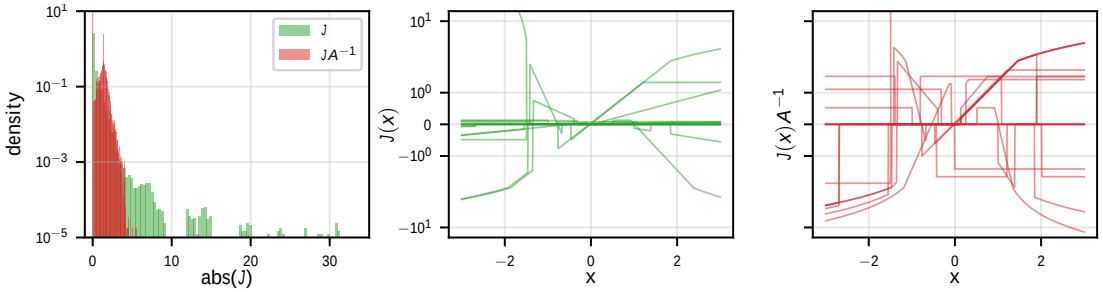

Figure 4: **Left:** Histogram of the training data's full Jacobian expansion $(J(x_i))_{i \leq n}$ for the NN from Figure 1, with and without g-prior scaling $A^{-1} = \text{diag}(I(w))^{-0.5}$. **Middle:** 15 randomly chosen Jacobian basis functions. **Right:** Same functions with g-prior scaling.

5. The Neural Network g-Prior

Regardless of normalisation layers, we observe different scales in the components of the Jacobian basis, e.g. see Figure 4. A single prior precision will be too restrictive for weights corresponding to small basis components, preventing these from influencing model predictions and error-bars while leaving weights corresponding to large components underspecified.

This issue is well studied in Bayesian linear models (Minka, 2000), with a standard solution being the use of Zellner's g-prior (Zellner, 1996). This is a Gaussian centred at 0 with precision the scaled Fisher information matrix $\pi_g(w) = \mathcal{N}(w; 0, g \cdot I^{-1})$. The scaling factor g gives name to the prior. For NNs the Fisher is often singular (Watanabe, 2007), making it ill-suited as a prior. We bypass this issue by diagonalising the prior

$$\pi_{\text{g-NN}}(w) \quad \exp\left(\frac{-1}{2g} w^T \text{diag}\left(I(w)\right) w\right), \tag{10}$$

with $\text{diag}(I(w)) = 1/n \sum_{i<n} \mathbb{E}_{p(y|f(x_i,w))}[(\nabla_w \log p(y|f(x_i,w))^2|_{w=w}]I$. This matrix will vary depending on the point w around which it is computed, resulting in an un-normalised density. The null space of eq. (10) only corresponds to weights which have no effect on the output (e.g. those corresponding to dead ReLUs). We drop these from our probabilistic model. The Laplace approximation reverts eq. (10) to $\mathcal{N}(w; 0, g \cdot \text{diag}(I(w))^{-1})$ with $I(w)$ matching the linearised NN's Fisher. Applying the proposed prior to our linearised model *can be interpreted as downscaling our Jacobian features by their second moment* while keeping our prior as an isotropic Gaussian of variance g. Details are in Appendix D.

6. Experiments

Our procedure is: 1) train NNs with standard methods, 2) place either an isotropic Gaussian $\mathcal{N}(0, \lambda I)$ or NN g-prior $\mathcal{N}(0, g \cdot \text{diag}(I(w))^{-1})$ over the parameters, 3) optimise the hyperparameters λ and g using either eq. (3) (with NN weights w) or eq. (9) (with linear weights v), and 4) predict using linearised Laplace. Implementation details are in Appendix E.

We first train a 2 hidden layer, 50 hidden unit, MLP with residual connections and *layer norm* on the "matern" 1d regression dataset from Antoran et al. (2020). Figure 2 shows that the likelihood is radially invariant while the prior is rotationally invariant. Once the network is linearised, the likelihood invariance only remains in the direction of w, as shown in Figure 3. Here we see that the norm of w is much larger than that of v. Consequently,

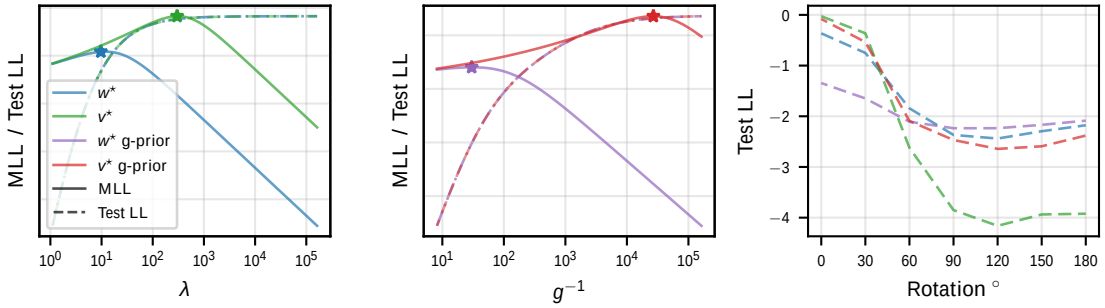

Figure 5: MNIST CNN MLL sweeps for normal and g-prior parameters λ and g. **Left:** With w the MLL eq. (3) optima corresponds to a smaller λ than with v eq. (9). The latter provides a higher test LL. For a given λ the test LL is the same in both cases since the predictive mean is the same. **Middle:** We observe the same behaviour when choosing g. **Right:** Test set evaluation, under several rotations, using optimised λ & g values.

optimising the NN MLL eq. (3) returns too small a precision and thus produces errorbars larger than the marginal variance of the targets in Figure 1. Choosing λ with the linearised model MLL eq. (9) produces much more sensible results qualitatively. When employing the g-prior, the errorbars are informed by a wider variety of basis functions and thus present a smoother, more sensible, shape.

Next, we train a 46k parameter *batch norm* Lenet-style CNN on MNIST. Figure 5 shows a much stronger test log-likelihood when hyperparameters are optimised with our proposed linearised MLL eq. (9) than when using eq. (3), for both isotropic and g priors. Given that in all settings, the predictive mean is the same across methods, we can attribute the increase in LL to better calibrated error-bars. Interestingly, our proposed model selection criteria eq. (9) tends to choose the smallest precision that maximises the test LL. This maximises uncertainty while not sacrificing in-distribution performance. We evaluate OOD uncertainty estimation on rotated digits. The hyperparameters found with the NN weights w perform well OOD at the cost of overestimating uncertainty in-distribution. Our proposed criteria combined with the isotropic Gaussian prior performs well in-distribution but is overconfident OOD. The more expressive g-prior, when combined with eq. (9), retains in-distribution performance while expressing increased uncertainty OOD.

7. Conclusion

We have highlighted and provided solutions to two pitfalls of the naive application of linearised Laplace to modern NNs. First, normalisation layers preclude finding an optima of the loss for a NN, or its tangent linear model, by optimising the loss function of the NN. 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 errorbars, we propose to also choose hyperparameters using this model's evidence. This can be done by solving a convex optimisation problem, much simpler than NN optimisation. Second, the scales of basis functions corresponding to the Jacobian entries for different weights can be very heterogeneous. We propose the extension of the g-prior to the NN setting. This empirical prior assigns a variance to each weight inversely proportional to the scale of its associated feature.

Acknowledgments

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. ED acknowledges funding from the EPSRC and Qualcomm. JUA acknowledges funding from the EPSRC and the Michael E. Fisher Studentship in Machine Learning. 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

Neculai Andrei. 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. URL https://www.sciencedirect.com/science/article/pii/S0377042708006523.

Javier Antoran, James Allingham, and José Miguel Hernández-Lobato. Depth uncertainty in neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 10620–10634. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/781877bda0783aac5f1cf765c128b437-Paper.pdf.

Javier Antoran, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. Getting a {clue}: A method for explaining uncertainty estimates. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=XSLF1XFq5h.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. URL http://arxiv.org/abs/1607.06450.

Riccardo Barbano, Javier Antoran, José Miguel Hernández-Lobato, and Bangti Jin. A probabilistic deep image prior over image space. In *Fourth Symposium on Advances in Approximate Bayesian Inference*, 2022. URL https://openreview.net/forum?id=qtFPfwJWowM.

Andy Brock, Soham De, Samuel L. Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. In Marina Meila and Tong Zhang, editors, *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*, pages 1059–1071. PMLR, 2021. URL http://proceedings.mlr.press/v139/brock21a.html.

Yongqiang Cai, Qianxiao Li, and Zuowei Shen. A quantitative analysis of the effect of batch normalization on gradient descent. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *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, pages 882–890. PMLR, 2019. URL http://proceedings.mlr.press/v97/cai19a.html.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux–effortless Bayesian deep learning. In M. Ranzato, A. Beygelzimer, P.S. Liang, J.W. Vaughan, and Y. Dauphin, editors, *Advances in Neural Information Processing Systems*, volume 34, 6–14 Dec 2021a. URL https://papers.nips.cc/paper/2021/hash/a7c9585703d275249f30a088cebba0ad-Abstract.html.

Erik Daxberger, Eric Nalisnick, James U Allingham, Javier Antoran, and Jose Miguel Hernandez-Lobato. Bayesian deep learning via subnetwork inference. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2510–2521. PMLR, 18–24 Jul 2021b. URL https://proceedings.mlr.press/v139/daxberger21a.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 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*, pages 770–778. IEEE Computer Society, 2016.

Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2164–2174, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/a0160709701140704575d499c997b6ca-Abstract.html.

Alexander Immer, Matthias Bauer, Vincent Fortuin, Gunnar Rätsch, and Mohammad Emtiyaz Khan. Scalable marginal likelihood estimation for model selection in deep learning. *CoRR*, abs/2104.04975, 2021a. URL https://arxiv.org/abs/2104.04975.

Alexander Immer, Maciej Korzepa, and Matthias Bauer. Improving predictions of bayesian neural nets via local linearization. In Arindam Banerjee and Kenji Fukumizu, editors, *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*, pages 703–711. PMLR, 2021b. URL http://proceedings.mlr.press/v130/immer21a.html.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *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*, pages 448–456. JMLR.org, 2015.

Mohammad Emtiyaz Khan, Alexander Immer, Ehsan Abedi, and Maciej Korzepa. Approximate inference turns deep networks into gaussian processes. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3088–3098, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/b3bbccd6c008e727785cb81b1aa08ac5-Abstract.html.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. 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*, pages 5436–5446. PMLR, 2020. URL http://proceedings.mlr.press/v119/kristiadi20a.html.

Zhiyuan Li, Kaifeng Lyu, and Sanjeev Arora. Reconciling modern deep learning with traditional optimization analyses: The intrinsic learning rate. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/a7453a5f026fb6831d68bdc9cb0edcae-Abstract.html.

Ekaterina Lobacheva, Maxim Kodryan, Nadezhda Chirkova, Andrey Malinin, and Dmitry P. Vetrov. On the periodic behavior of neural network training with batch normalization and weight decay. *CoRR*, abs/2106.15739, 2021. URL https://arxiv.org/abs/2106.15739.

David John Cameron Mackay. *Bayesian Methods for Adaptive Models*. PhD thesis, USA, 1992. UMI Order No. GAX92-32200.

James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

Tom Minka. Bayesian linear regression. July 2000. URL https://www.microsoft.com/en-us/research/publication/bayesian-linear-regression/.

Twan van Laarhoven. L2 regularization versus batch and weight normalization. *CoRR*, abs/1706.05350, 2017. URL http://arxiv.org/abs/1706.05350.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Sumio Watanabe. Almost all learning machines are singular. In *Proceedings of the IEEE Symposium on Foundations of Computational Intelligence, FOCI 2007, part of the IEEE Symposium Series on Computational Intelligence 2007, Honolulu, Hawaii, USA, 1-5 April 2007*, pages 383–388. IEEE, 2007. doi: 10.1109/FOCI.2007.371500. URL https://doi.org/10.1109/FOCI.2007.371500.

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

Arnold Zellner. Models, prior information, and bayesian analysis. *Journal of Econometrics*, 75(1):51–68, 1996.

Appendix A. On Invariances Introduced by Normalisation Layers

We relate eq. (4) to common normalisation layers. For some input $z \in \mathbb{R}^{d_z}$ these apply the function

$$\frac{z - \mathbb{E}[z]}{\sqrt{\mathrm{Var}(z)}}\gamma + \beta$$

where the expectation and variance are estimated empirically and $\gamma \in \mathbb{R}$, $\beta \in \mathbb{R}$. We have invariance to scaling the input by some constant $\kappa \in \mathbb{R}$

$$\frac{\kappa z - \mathbb{E}[\kappa z]}{\sqrt{\mathrm{Var}(\kappa z)}}\gamma + \beta = \frac{\kappa(z - \mathbb{E}[z])}{\kappa\sqrt{\mathrm{Var}(z)}}\gamma + \beta = \frac{z - \mathbb{E}[z]}{\sqrt{\mathrm{Var}(z)}}\gamma + \beta.$$

For *batch norm*, the input to the above function z, corresponds to the different values of a neural activation across a mini-batch of data. As a result, normalisation groups correspond to the fan-in weights of each node. For *layer norm*, z corresponds to a layer's activations for a single input. Here, the normalisation group is the complete set of layer weights. *Group norm* applies the same operation as layer norm but across a subset of a convolutional layer's channels. Here, the normalisation group is the subset of the previous layer's weights which acts on any of the output channels that fall into a specific group.

Appendix B. Derivation and Analysis of Algorithm 1

We denote NN Jacobians as $J(x) =: \nabla_w f(x, w)|_{w=w} \in \mathbb{R}^{d_y \times d_w}$ and a linear model weight vector $v \in \mathbb{R}^{d_w}$. d_y is the output size. Consider the linear model $J(x)v$ where we wish to optimise v according to the objective $\log p(y|J(x)v)$. Expanding the expression of the gradient using the chain rule

$$\nabla_v \log p(y|J(x)v) = \nabla_{\hat{y}} p(y|\hat{y})|_{\hat{y}=J(x)v} J(x)$$

we see that it is equal to the gradient of the loss projected onto the weights through the Jacobian. We first need to evaluate our basis function linear model $\hat{y} = J(x)v$. This Jacobian vector product can be computed with forward mode automatic differentiation or finite differences (FD). We rely on finite differences because it is slightly cheaper to compute. Specifically, we employ the method of Andrei (2009) to select the optimal step size. We empirically evaluate this approach in Figure 6, finding it to return the near-optimal step-size for a wide range of queries. We then evaluate the loss gradient at the linear model output, $g_y = \nabla_{\hat{y}} p(y|\hat{y})|_{\hat{y}=J(x)v}$. This can often be done in closed form. Finally, we project onto the weights using backward mode automatic differentiation $g_w = g_y^T J(x) = \nabla_w (g_y^T f(x, w))|_{w=w}$.

For the small MLP used in the toy experiments from Section 6, it is tractable to compute the Jacobians for the full training set. In this setting, we can evaluate the fidelity of the gradient update computed with algorithm 1 relative to exact gradient descent (GD). In Figure 7, we show that the proposed FD based method tracks the GD trajectory very closely. The gradient bias, computed in terms of rmse across all weights, stays below $2e-2$ during the whole optimisation. The bias in the weights accumulates throughout training, reaching a maximum of ≈ 0.015 at convergence.

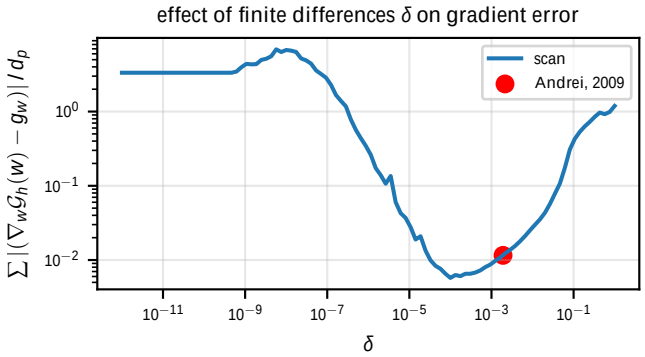

Figure 6: Average per-weight absolute gradient error obtained when computing $J(x)v$ with two-sided finite differences in step 2 of algorithm 1 with different step sizes δ. The method of Andrei (2009) provides almost optimal step sizes.

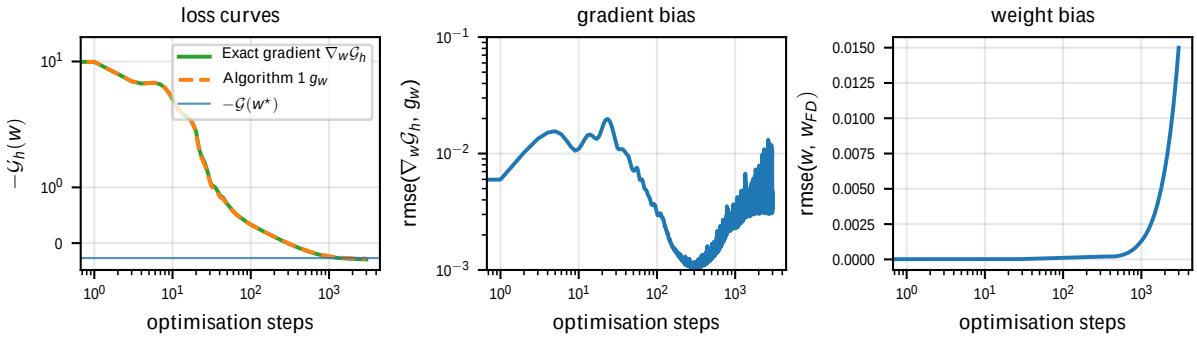

Figure 7: **Left:** Optimisation trajectories for exact gradient descent and the proposed method: algorithm 1. **Middle:** Gradient bias throughout training in terms of rmse across all weights. **Right:** Weight bias throughout training in terms of rmse across all weights.

In Figure 8, we compare the fits and weight histograms obtained when optimising the linear model corresponding to the MLP from Section 6 with algorithm 1 and with the closed form solution for Gaussian linear regression. Both fits agree seemingly perfectly in the range of the data, with some slight disagreement in the extrapolation regime. The weight histograms overlap almost perfectly in the body, with some outliers being placed in different locations. This suggests that the MLL values obtained with eq. (9) when finding v with algorithm 1 will be a good approximation to the true linearised model MLL.

Appendix C. Additional Plots

In Figure 9 we compare the fits obtained on our toy problem when employing the NN function $f(x, w)$ and the linearised model $h(x, v)$. In the latter case we compare both an isotropic Gaussian prior and the a g-prior. All three methods return very similar mean predictions, with the most notable difference being the g-prior's additional non-smoothness in the out-of-distribution regime. This is due to the g-prior allowing for more basis functions to contribute the the mean prediction. The histogram plot shows that w presents a both

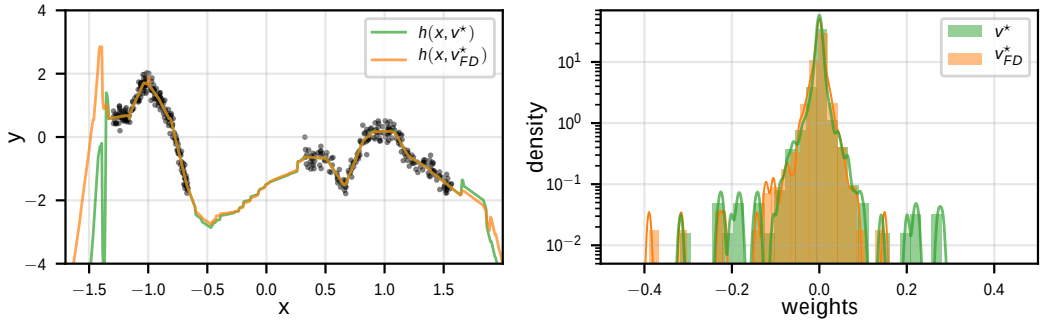

Figure 8: **Left:** Fits obtained with the exact MAP setting of a linearised MLP and with our proposed optimisation algorithm 1 on the toy task presented in Section 6. **Right:** Histogram displaying the MAP weights computed with each method.

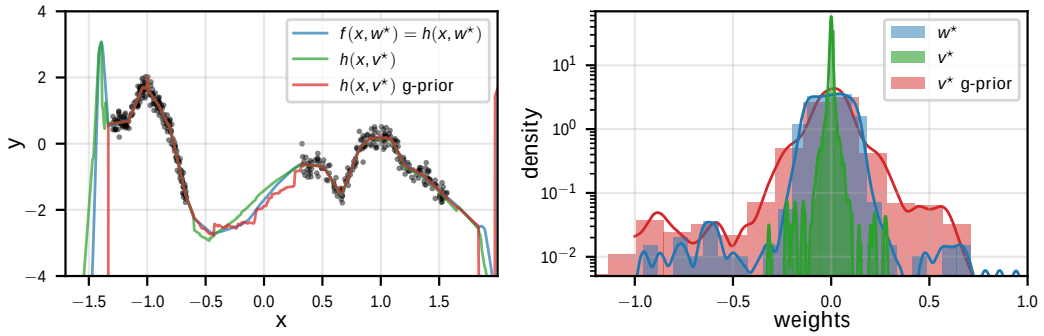

Figure 9: **Left:** Fits obtained with an MLP $f(x, w)$ and its linearised model $h(x, v)$ on the toy task presented in Section 6. For the latter we compare an isotropic Gaussian prior and with the g-prior. **Right:** Histogram displaying MAP weights for all 3 settings.

wider and heavier tailed distribution than v. This explains why choosing hyperparameters with eq. (3) leads to excessively wide error bars in Figure 1. The weights obtained when using the g-prior present a wider distribution than w but show less heavy tails.

Figure 10 shows the weight histogram for the CNN used in Section 6. Its architecture is described in Appendix E. The NN weights w are significantly larger than those of the linearised model v. As a result, the MLL that uses w favours smaller prior precisions which result in strong underconfidence in-distribution, as found in Section 6.

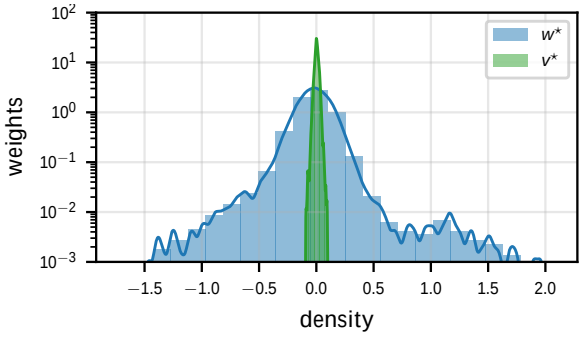

Figure 10: Histogram displaying MAP weights for our CNN (described in Appendix E) w and its linearised counterpart v .

Appendix D. Feature Normalisation and the Diagonal g-Prior

In this appendix, we provide some intuition for the effects of the diagonal g-prior on the linearised NN model. We stack the features $(J(x_i))_{i \leq n}$, into the tensor $\boldsymbol{J} \quad \mathbb{R}^{n \times d_y \times d_w}$. We will suggestively refer to the tensor that has the first and third indexing dimensions flipped as $\boldsymbol{J}^T \quad \mathbb{R}^{d_w \times d_y \times n}$ for notational convenience. The log-likelihood Hessian, stacked across observations is $H \quad \mathbb{R}^{n \times d_y \times d_y}$. For exponential family distributions, the Fisher $I \quad \mathbb{R}^{d_w \times d_w}$ matches the GGN matrix (Martens, 2014)

$$I = \sum_i (\boldsymbol{J}_i) \quad H_i \boldsymbol{J}_i \tag{11}$$

and thus the NN g-prior precision is

$$g^{-1}\mathrm{diag}(\sum_i \boldsymbol{J}_i \quad H_i \boldsymbol{J}_i) I = g^{-1} A^T A. \tag{12}$$

$A \quad \mathbb{R}^{d_w \times d_w}$ is a diagonal matrix which will act as a feature normaliser $\boldsymbol{J}^{\mathrm{s}} = \boldsymbol{J} A^{-1}$. The entries of A are

$$A^2_{k,k} = \frac{1}{n} \sum_i \sum_l \sum_j \boldsymbol{J}_{i,l,k} H_{i,l,j} \boldsymbol{J}_{i,j,k}, \tag{13}$$

where H regulates linear interactions across output dimensions and observations.

D.1. Parameter Optima, Linearised Laplace Predictive Distribution and Laplace Marginal Likelihood induced by Feature Normalisation and Diagonal g-Prior

Consider a generalised linear model $h(x, w)$, which we optimise with the loss function $G(\hat{y}) = \sum_{i \leq n} \log p(y_i | \hat{y})$. We consider 2 scenarios. In the first we choose the regulariser based on the diagonal g-prior $\log \pi_g(v) \quad -0.5 g^{-1} \cdot v^T A^T A v$. In the second we choose an isotropic regulariser $\log \pi(v) \quad g^{-1} \cdot v^T v$ but we scale our features as $\boldsymbol{J}^{\mathrm{s}} = \boldsymbol{J} A^{-1}$. We refer to the former as the g-prior model and to the latter as the scaled model.

Parameter Optima An optima of the parameters v satisfies

$$(\quad {}_{v}G(h(\{x_i\}_{i=0}^{n}, v)) - \quad {}_{v}\log\pi(v))|_{v=v^*} = 0. \tag{14}$$

We first apply the chain rule to the loss gradient ${}_{w}G(w)|_{w=w^*} = {}_{\hat{y}}G(\hat{y})|_{\hat{y}=w^*}\ {}_{J}J$. In the g-prior setting, this is

$$(\quad {}_{\hat{y}}G(\hat{y}))|_{\hat{y}=v_g J^T} J + g^{-1} \cdot A^T A v_g = 0 \tag{15}$$

and we have denoted the stationary point v_g. For the scaled model, the stationary point is

$$(\quad {}_{\hat{y}}G(\hat{y}))|_{\hat{y}=v_s A^{-1} J^T} J A^{-1} + g^{-1} \cdot v_s = (\quad {}_{\hat{y}}G(\hat{y}))|_{\hat{y}=v_s A^{-1} J^T} J + g^{-1} \cdot A v_s = 0 \tag{16}$$

with stationary point v_s. Combining both eqs. (15) and (16) we get that the scaled model weights are just the scaling matrix applied to the g-prior weights $A^{-1} v_s = v_g$.

We now show this in the special case of the Gaussian likelihood case, with precision H,

$$v_g = \Big(\sum_i (J_i)^T H_i J_i + g^{-1} \cdot A^T A\Big)^{-1} \sum_j (J_j)^T y =$$
$$A^{-1}\Big(\sum_i A^{-1}(J_i)^T H_i J_i A^{-1} + g^{-1} \cdot I\Big)^{-1} \sum_j A^{-1}(J_j)^T y = A^{-1} v_s \tag{17}$$

Linearised Laplace Predictive The predictive mean for both the g-prior model and scaled model is the same:

$$\mathsf{E}_{p_g(v|(x_i,y_i)_{i=1}^n)}[Jv] = J(x)v_g = J(x)A^{-1}v_s = \mathsf{E}_{p_s(v|(x_i,y_i)_{i=1}^n)}[vA^{-1}J] \tag{18}$$

The Laplace predictive covariance is also the same:

$$J(x)\Big(\sum_i (J_i)^T H_i J_i + g^{-1} \cdot A^T A\Big)^{-1} J^T(x)$$
$$= A^{-1} J(x)\Big(\sum_i A^{-1}(J_i)^T H_i J_i A^{-1} + g^{-1} \cdot\Big)^{-1} J^T(x) A^{-1} \tag{19}$$

Linearised Laplace MLL This quantity also matches for the g-prior model and scaled model:

$$-g^{-1} \cdot v_g{}^T A^T A v_g - \log \frac{\det(\sum_i^n (J_i)^T H_i J_i + g^{-1} \cdot A^T A)}{\det(g^{-1} \cdot A^T A)}$$
$$= g^{-1} \cdot v_s{}^T A^{-T} A^T A A^{-1} v_s - \log \frac{\det(A^{-1}) \det(\sum_i^n (J_i)^T H_i J_i + g^{-1} \cdot A^T A) \det(A^{-1})}{\det(g^{-1} \cdot I)}$$
$$= g^{-1} \cdot v_s{}^T v_s - \log \frac{\det(\sum_i^n A^{-1}(J_i)^T H_i J_i A^{-1} + g^{-1} I)}{\det(g^{-1} \cdot I)}. \tag{20}$$

We have dropped the data fit terms and other constants as these match for the g-prior and scaled model since both models make the same mean predictions eq. (18).

D.2. What type of Feature Normalisation does the Diagonal g-Prior Induce?

In this subsection we discuss the similarity between the scaled model, induced by use of the g-prior and a generalised linear model in which the inputs have been scaled such that all inputs share some summary statistic (e.g. 0 mean, unit variance, etc). We make the assumption that our data is iid and thus the likelihood function is factorised across datapoints. Let's first consider the case where our log-likelihood is homoscedastic and factorised across output dimensions $H_i = \kappa \cdot I \quad i \leq n$, where $\kappa \quad \mathsf{R}_+$ is the likelihood precision. Here we have that the *scaling matrix matches the empirically computed second moment of each basis function in our expansion*

$$A^2_{k,k} = \frac{\kappa}{n} \sum_i \sum_l J^2_{i,l,k} = \kappa \sum_l \mathsf{E}_{\hat{p}(x)}[J^2_l(x)], \tag{21}$$

summed across output dimensions and scaled by the noise precision. In fact, the constant κ can be absorbed into g. For one output dimension $d_y = 1$, this leaves us with a scaling procedure that matches commonly used standard deviation normalisation for the case of 0-mean features. Whether our Jacobian features are 0-mean will depend on if the linearisation point w is stationary. In the multi-response setting $d_y > 1$, the standardisation for the feature corresponding to each weight is proportional to the sum of the second moments of the gradients of each output dimension with respect to the weight. For a heteroscedastic likelihood function factorised across output dimensions $H_i = \kappa_i \cdot I \quad i < n$, where $\kappa \quad \mathsf{R}^n_+$ we have

$$A^2_{k,k} = \frac{1}{n} \sum_i \sum_l \kappa_i J^2_{i,l,k}. \tag{22}$$

Now different datapoints contribute to the scaling factor proportionally to their noise precision. Finally we consider the case where our likelihood function is heteroscedastic and does not factorise across output dimensions, like is the case for the softmax-categorical. Here H_i is full rank. The entries of our scaling matrix are given by eq. (13).

Appendix E. Image Classification Experimental Setup

For the image classification experiments, we employ a CNN 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 a convolution layer, a ReLU activation, and a batch norm layer, in that order. 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. Table 1 shows the sizes of the filters and number of parameters. These where chosen to create a model as large as possible while keeping full-covariance Laplace inference tractable on one A100 GPU.

The NN weights w 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 multi-step LR scheduler with a decay rate of 0.1 applied at epochs 40 and 70.

Table 1: Architecture parameters for CNN used in experiments.

	Conv1 Filters	Conv2 Filters	Conv3 Filters	Params.	Hessain Size
CNN	42	48	60	46 024	15.68 GB

The linear weights v are learnt using SGD but with the gradient calculated via algorithm 1. We use a learning rate of 1×10^{-4} and train for 100 epochs. We set the weight decay value to 1×10^{-4} when using isotropic priors and to 1×10^{-1} when scaling features according to the g-prior. The latter choice is made due to the model with scaled features having a larger effective dimensionality and thus needing stronger regularisation. We do not use momentum or learning rate decay.