A EXTENDED RELATED WORK

Bayesian Gaussian linear models This work builds on the rich literature of Bayesian linear regression (Gull, 1989; Bishop, 2006; Rasmussen & Williams, 2006). Specifically, we present a stochastic approximation to the iterative algorithm for hyperparameter selection introduced by (Mackay, 1992) and extended by Tipping (2001); Tipping & Faul (2003); Wipf & Nagarajan (2007); Antorán et al. (2022c). Analytical tractability makes linear models ubiquitous in machine learning, with applications in genomics (Runcie et al., 2021), reinforcement learning (Ash et al., 2022), and pandemic modelling (Nicholson et al., 2022), among others. Alas, linear models are held back by a cost of inference cubic in the number of parameters when expressed in primal form, or cubic in the number of observations for the dual (i.e. kernelised or Gaussian Process) form. Additionally, for non-Gaussian likelihoods, e.g. in classification, inference is no longer closed form. The most common approximations used in these settings are Laplace's method (Mackay, 1992) and variational inference (Hensman et al., 2013). Khan et al. (2019) and Adam et al. (2021) show that every Gaussian approximation corresponds to the true posterior of a surrogate regression problem with the same features, a fact which we use in this work to apply sample-then-optimise to Laplace posteriors.

Sample-then-optimise Papandreou & Yuille (2010); de G. Matthews et al. (2017) phrase sampling from a conjugate Gaussian-linear model as solving a perturbed quadratic optimisation problem. This method has been applied for uncertainty estimation in non-linearised NNs by Osband et al. (2018; 2021), and Pearce et al. (2020), although in this setting it does not draw exact posterior samples. In this work, we show sample-then-optimise to be the primal form of Matheron's rule (Journel & Huijbregts, 1978; Hoffman & Ribak, 1991), a method for updating jointly Gaussian samples into conditional samples, which was recently repopularised by Wilson et al. (2020).

Linearised neural networks Introduced by Mackay (1992), these are an approximation yielding closed-form errorbars for Laplace posteriors. Lawrence (2000) and Ritter et al. (2018) found the Laplace approximation to underperform without the linearisation step. Khan et al. (2019) and Immer et al. (2021b) re-popularised the linearisation step by showing that it improves the quality of uncertainty estimates. Kristiadi et al. (2020) show that the Laplace approximation is sufficient to resolve certain pathologies of point-estimated NNs' predictions. Immer et al. (2021a) and Antorán et al. (2022a;c) explore the linear model's evidence for model selection. Daxberger et al. (2021b) and Maddox et al. (2021) introduce subnetwork and finite differences approaches, respectively, for faster inference with the linearised model. This line of work is intimately related to the neural tangent kernel (Jacot et al., 2018; Lee et al., 2019; Novak et al., 2020) in which NNs are linearised at initialisation.

The g-prior, originally introduced by Zellner (1986), consists of a centred Gaussian with covariance matching the inverse of the Fisher information matrix. Resultantly, the g-prior ensures inferences are independent of the units of measurement of the covariates (Minka, 2000). Since then, it has extensively used in the context of model selection for generalised linear models (Liang et al., 2008; Bové & Held, 2011; Baragatti & Pommeret, 2012). In the large-scale setting, we overcome the computational intractability of the Fisher by diagonalising the g-prior while preserving its invariance property.

B MODEL EVIDENCE LOWER BOUND AND THE EFFECTIVE DIMENSION

B.1 EQUIVALENT FORMULATIONS OF EFFECTIVE DIMENSION

We begin by relating two standard forms of effective dimension, which we use throughout. Starting with the form standard in the kernel-based literature (that without an explicit d' dependence),

$$\gamma = \text{Tr}\left\{(A+M)^{-1}M\right\} = \text{Tr}\left\{(I+A^{-1}M)^{-1}A^{-1}M\right\} = \text{Tr}\left\{I-(I+A^{-1}M)^{-1}\right\} \tag{18}$$

$$= d' - \text{Tr}\left\{A(A+M)^{-1}\right\}, \tag{19}$$

we have arrived at the form used within the finite-dimensional linear modelling literature (Mackay, 1992; Wipf & Nagarajan, 2007; Maddox et al., 2020).

B.2 DERIVATION OF $\mathcal{M}$ AS A LOWER BOUND ON THE MODEL EVIDENCE

Let p_θ be the Lebesgue density of $\mathcal{N}(\Phi\theta, B^{-1})$, $P = \mathcal{N}(0, A'^{-1})$ and $Q = \mathcal{N}(\bar{\theta}, (M + A')^{-1})$. Then,

$$\log p(Y; A') = \log \int p_\theta(Y) dP = \log \int p_\theta(Y) \frac{dP}{dQ} dQ \geq \int \log \left[p_\theta(Y) \frac{dP}{dQ} \right] dQ \tag{20}$$

$$= \int \log p_\theta(Y) dQ - \mathrm{D}(Q||P). \tag{21}$$

where D denotes the KL-divergence. Starting with the first term,

$$\int \log p_\theta(Y) dQ = \frac{1}{2} \int -n \log 2\pi + \log \mathrm{detB} - (Y - \Phi\theta)^T \mathrm{B}(Y - \Phi\theta) dQ \tag{22}$$

$$= \frac{1}{2} \left[-n \log 2\pi + \log \mathrm{detB} \right] - \frac{1}{2} \int (Y - \Phi\theta)^T \mathrm{B}(Y - \Phi\theta) dQ, \tag{23}$$

and expanding the quadratic form,

$$\int (Y - \Phi\theta)^T \mathrm{B}(Y - \Phi\theta) dQ = Y^T \mathrm{B} Y - 2Y^T \mathrm{B}\Phi \int \theta dQ + \int \theta^T \Phi^T \mathrm{B}\Phi\theta dQ \tag{24}$$

$$= Y^T \mathrm{B} Y - 2Y^T \mathrm{B}\Phi\bar{\theta} + \int \theta^T M \theta dQ. \tag{25}$$

To handle the final integral, consider that

$$\gamma = \mathrm{Tr}\left\{ M(M + A')^{-1} \right\} \tag{26}$$

$$= \mathrm{Tr}\left\{ M \int (\theta - \bar{\theta})(\theta - \bar{\theta})^T dQ \right\} \tag{27}$$

$$= -\mathrm{Tr}\left\{ M\bar{\theta}\bar{\theta}^T \right\} + \mathrm{Tr}\left\{ M \int \theta\theta^T dQ \right\} \tag{28}$$

$$= -\bar{\theta}^T M \bar{\theta} + \int \theta^T M \theta dQ, \tag{29}$$

and thus

$$\int \log p_\theta(Y) dQ = \frac{1}{2} \left[\log \mathrm{detB} - n \log 2\pi - (Y - \Phi\bar{\theta})^T \mathrm{B}(Y - \Phi\bar{\theta}) - \gamma \right] \tag{30}$$

$$= \log p_{\bar{\theta}}(Y) - \frac{1}{2}\gamma. \tag{31}$$

The KL between two multivariate Gaussians is a standard result, yielding

$$\mathrm{D}(Q||P) = \frac{1}{2} \left[-\log \mathrm{det} A' + \log \mathrm{det}(M + A') - d' + \bar{\theta}^T A' \bar{\theta} + \mathrm{Tr}\left\{ A'(M + A')^{-1} \right\} \right] \tag{32}$$

$$= \frac{1}{2} \left[-\log \mathrm{det} A' + \log \mathrm{det}(M + A') + \bar{\theta}^T A' \bar{\theta} - \gamma \right], \tag{33}$$

where we used that $\gamma = d' - \mathrm{Tr}\left\{ A'(M + A')^{-1} \right\}$.

Putting together (31) and (33), we obtain

$$\log p(Y; A') \geq \log p_{\bar{\theta}}(Y) - \frac{1}{2} \log \mathrm{det}(A'^{-1} M + I) - \frac{1}{2} \|\bar{\theta}\|^2_{A'} = \mathcal{M}(A'), \tag{34}$$

which is the stated result up to taking $C = \log p_{\bar{\theta}}(Y)$.

B.3 FIRST ORDER OPTIMALITY CONDITION FOR $\mathcal{M}$

Consider the derivative of $\mathcal{M}$. We have,

$$\nabla \mathcal{M}(A) = -\frac{1}{2} \left[\nabla \|\bar{\theta}\|^2_A + \nabla \log \mathrm{det}(A + M) - \nabla \log \mathrm{det} A \right], \tag{35}$$

where we expanded $\log\det(I + A^{-1}M) = \log\det(A + M) - \log\det A$. Taking the respective derivatives and setting equal to zero at A, this leads to the condition

$$\bar{\theta}\bar{\theta}^T = (I - (I + A^{-1}M)^{-1})A^{-1}. \tag{36}$$

Post-multiplying by A and applying the push-through identity, we obtain

$$\bar{\theta}\bar{\theta}^T A = M(A + M)^{-1}. \tag{37}$$

For the above to hold, it is necessary that the traces of both sides are equal. Thus,

$$\|\bar{\theta}\|_A^2 = \mathrm{Tr}\{\bar{\theta}\bar{\theta}^T A\} = \mathrm{Tr}\{M(A + M)^{-1}\} = \gamma, \tag{38}$$

which is the stated first order optimality condition, up to a cyclic permutation.

B.4 M-step for feature-wise regularisation strengths

We can leverage the primal form expression for the effective dimension given in Appendix B.1 to extend the above first order optimality condition to the feature-wise regulariser setting.

Consider a sub-vector of our weight vector contiguous between the ith and jth weights written as $\bar{\theta}_{i:j}$. Note that we only choose contiguous weights for notational convenience but it is not necessary to do so in general.

The first order condition from Appendix B.3 is satisfied if for any i, j with $i < j$ we have

$$\alpha\|\bar{\theta}_{i:j}\|^2 = |j - i| - \alpha\sum_{k=i}^{j}[(A + M)^{-1}]_{kk} \coloneqq \gamma_{i:j}, \tag{39}$$

and thus we may update the regulariser for each separate sub-vector as $\alpha = \gamma_{i:j}/\|\bar{\theta}_{i:j}\|^2$.

C Analysis of losses and loss gradient estimator variances

C.1 On loss minima

The losses L and L' are strictly convex, thus to confirm they have the same unique minimum, it suffices to consider the respective first order optimality conditions, $\nabla L(\zeta) = 0$ and $\nabla L'(\zeta') = 0$. We have,

$$\nabla L(\zeta) = \Phi^T \mathrm{B}(\Phi\zeta - \mathcal{E}) + A(\zeta - \theta^0), \tag{40}$$

and

$$\nabla L'(\zeta') = \Phi^T \mathrm{B}\Phi\zeta' + A(\zeta' - A^{-1}\Phi^T \mathrm{B}\mathcal{E} - \theta^0) \tag{41}$$

$$= \Phi^T \mathrm{B}(\Phi\zeta' - \mathcal{E}) + A(\zeta' - \theta^0) \tag{42}$$

Thus $\zeta = \zeta'$ almost surely. Moreover, $L'(z) = L(z) + C$ for all z, for C a constant independent of z.

To determine the distribution of ζ, note that it is a linear transformation of zero-mean Gaussian random variables, and thus itself a zero-mean Gaussian random variable. Rearranging the first order optimality condition, we find that

$$\zeta = H^{-1}(\Phi^T \mathrm{B}\mathcal{E} + A\theta^0). \tag{43}$$

Thus

$$\mathbb{E}[\zeta\zeta^T] = H^{-1}\mathbb{E}[(\Phi^T \mathrm{B}\mathcal{E} + A\theta^0)(\Phi^T \mathrm{B}\mathcal{E} + A\theta^0)^T]H^{-1} \tag{44}$$

$$= H^{-1}\left(\Phi^T \mathrm{B}\mathbb{E}[\mathcal{E}\mathcal{E}^T]\mathrm{B}\Phi + A\mathbb{E}[\theta^0\theta^0]A + 2\Phi^T \mathrm{B}\mathbb{E}[\mathcal{E}(\theta^0)^T]A\right)H^{-1} \tag{45}$$

$$= H^{-1}(\Phi^T \mathrm{B}\Phi + A)H^{-1} = H^{-1}HH^{-1} = H^{-1}. \tag{46}$$

And so $\zeta \sim \mathcal{N}(0, H^{-1}) = \Pi^0$ as claimed.

C.2 LOSS GRADIENT VARIANCE CONDITION

Taking $j \sim \text{Unif}(\{1, \dots, n\})$, the gradient estimators for the data-dependent terms of L and L' are

$$\hat{g} = n\nabla\|\phi(x_j)z - \varepsilon_j\|^2_{B_j} = n\phi(x_j)^T B_j(\phi(x_j)z - \varepsilon_j) \tag{47}$$

and

$$\hat{g}' = n\nabla\|\phi(x_j)z\|^2_{B_j} = n\phi(x_j)^T B_j\phi(x_j)z, \tag{48}$$

respectively. Their variances are related as

$$\begin{aligned}
\text{Var}(\hat{g}) &= \text{Var}(n\phi(x_j)^T B_j(\phi(x_j)z - \varepsilon_j)) && (49)\\
&= \text{Var}(n\phi(x_j)^T B_j\phi(x_j)z) + \text{Var}(n\phi(x_j)^T B_j\varepsilon_j) \\
&\quad - 2\text{Cov}(n\phi(x_j)^T B_j\phi(x_j)z,\ n\phi(x_j)^T B_j\varepsilon_j) && (50)\\
&= \text{Var}(\hat{g}') + \text{Var}(n\phi(x_j)^T B_j\varepsilon_j) - 2\text{Cov}(n\phi(x_j)^T B_j\phi(x_j)z,\ n\phi(x_j)^T B_j\varepsilon_j) && (51)
\end{aligned}$$

Evaluating the variance and covariance, we have

$$\text{Var}\left(n\phi(x_j)^T B_j\varepsilon_j\right) = n\text{Var}(\Phi^T \text{B}\mathcal{E}) \tag{52}$$

and

$$\text{Cov}(n\phi(x_j)^T B_j\phi(x_j)z,\ n\phi(x_j)^T B_j\varepsilon_j) = n\text{Cov}(\Phi^T \text{B}\Phi z,\ \Phi^T \text{B}\mathcal{E}), \tag{53}$$

and thus

$$\text{Var}\hat{g} - \text{Var}\hat{g}' = n\left[\text{Var}(\Phi^T \text{B}\mathcal{E}) - 2\text{Cov}(\Phi^T \text{B}\Phi z, \Phi^T \text{B}\mathcal{E})\right] =: n\Delta. \tag{54}$$

C.3 CONDITION AT CONVERGENCE

Now consider $\text{Tr}\,\Delta$ for $z = \zeta \sim \Pi^0$, the optimum of both L and L'. From the first order optimality condition,

$$\zeta = H^{-1}(\Phi^T \text{B}\mathcal{E} + A\theta^0). \tag{55}$$

Proceeding to rearrange the condition at $z = \zeta$,

$$\begin{aligned}
\text{Tr}\,\Delta &= \text{Tr}\left\{\mathbb{E}\Phi^T \text{B}\mathcal{E}(\Phi^T \text{B}\mathcal{E} - 2\Phi^T \text{B}\Phi\zeta)^T\right\} && (56)\\
&= \text{Tr}\left\{\mathbb{E}\Phi^T \text{B}\mathcal{E}(\Phi^T \text{B}\mathcal{E} - 2\Phi^T \text{B}\Phi H^{-1}(\Phi^T \text{B}\mathcal{E} + A\theta^0))^T\right\} && (57)\\
&= \text{Tr}\left\{\mathbb{E}\Phi^T \text{B}\mathcal{E}(\Phi^T \text{B}\mathcal{E} - 2\Phi^T \text{B}\Phi H^{-1}(\Phi^T \text{B}\mathcal{E} + A\mathbb{E}[\theta^0]))^T\right\} && (58)\\
&= \text{Tr}\left\{\mathbb{E}\Phi^T \text{B}\mathcal{E}(\Phi^T \text{B}\mathcal{E} - 2\Phi^T \text{B}\Phi H^{-1}\Phi^T \text{B}\mathcal{E})^T\right\}, && (59)\\
&= \text{Tr}\left\{\Phi^T \text{B}\mathbb{E}[\mathcal{E}\mathcal{E}^T](\text{B}\Phi - 2\text{B}\Phi H^{-1}\Phi^T \text{B}\Phi)\right\}, && (60)\\
&= \text{Tr}\left\{\Phi^T \text{B}\Phi(I - 2H^{-1}\Phi^T \text{B}\Phi)\right\} && (61)
\end{aligned}$$

where we substituted in the definition of ζ, then used that $\mathcal{E}$ and θ^0 are independent, and that $\mathbb{E}[\theta^0] = 0$, and finally that $\mathbb{E}[\mathcal{E}\mathcal{E}^T] = \text{B}^{-1}$.

Writing $M = \Phi^T \text{B}\Phi$ and recalling that $H = (M + A)$, we have

$$\begin{aligned}
\text{Tr}\,\Delta &= \text{Tr}\left\{M(I - 2(M + A)^{-1}M)\right\}, && (62)\\
&= \text{Tr}\left\{M(I - 2(A^{-1}M + I)^{-1}A^{-1}M)\right\}, && (63)\\
&= \text{Tr}\left\{M(I - 2(I - (A^{-1}M + I)^{-1})\right\}, && (64)\\
&= \text{Tr}\left\{M(-I + 2(M + A)^{-1}A)\right\} && (65)\\
&= -\text{Tr}\left\{M\right\} + 2\text{Tr}\left\{M(M + A)^{-1}A\right\}, && (66)
\end{aligned}$$

where we have used that $(A^{-1}M + I)^{-1}A^{-1}M = I - (A^{-1}M + I)^{-1}$ for the fourth equality. Now consider the isotropic prior case $A = \alpha I$ and recall the effective dimension is written as $\gamma = \text{Tr}\left\{M(M + A)^{-1}\right\}$. The above implies $\text{Tr}\,\Delta > 0$ if and only if $2\alpha\gamma > \text{Tr}\,\Phi^T \text{B}\Phi$.

C.4 ANALYSING CONDITION AT CONVERGENCE

To gain some intuition for the condition at convergence, denote by $\lambda_1, \ldots, \lambda_{d'}$ the eigenvalues of M (with multiplicity). We can use these to restate the condition as

$$2\alpha\gamma = 2\alpha \sum_{j=1}^{d'} \frac{\lambda_j}{\lambda_j + \alpha} > \sum_{j=1}^{d'} \lambda_j = \mathrm{Tr}\{M\}. \tag{67}$$

This formulation of effective dimension gives an interpretation of a soft count of the number of dimensions for which λ_j is larger than α; in that sense, λ_j measures how well determined the corresponding dimension of the weight vector θ is by the observed data. From here, note that

$$\frac{2\alpha\lambda_j}{\lambda_j + \alpha} > \min\{\lambda_j, \alpha\}, \tag{68}$$

and thus it is sufficient for $\mathrm{Tr}\,\Delta > 0$ to hold at convergence that $\alpha > \lambda_j$ for all j (but, of course, not necessary), yielding the intuition that L' is preferred when the problem is heavily regularised.

D DUAL FORM OF THE SAMPLE-THEN-OPTIMISE LOSS: MATHERON'S RULE

Both losses L and L' result in a random variable $\zeta \sim \Pi^0$ given by

$$\zeta = H^{-1}(\Phi^T \mathrm{B}\mathcal{E} + A\theta^0). \tag{69}$$

Recalling that $H = A + \Phi^T \mathrm{B}\Phi$ and using the push-through identity, we can express ζ equivalently as

$$\begin{aligned}
\zeta &= H^{-1}((H - \Phi^T \mathrm{B}\Phi)\theta^0 + \Phi^T \mathrm{B}\mathcal{E}) && (70)\\
&= \theta^0 + H^{-1}\Phi^T \mathrm{B}(\mathcal{E} - \Phi\theta^0) && (71)\\
&= \theta^0 + A^{-1}(I + \Phi^T \mathrm{B}\Phi A^{-1})^{-1}\Phi^T \mathrm{B}(\mathcal{E} - \Phi\theta^0) && (72)\\
&= \theta^0 + A^{-1}\Phi^T \mathrm{B}(I + \Phi A^{-1}\Phi^T \mathrm{B})^{-1}(\mathcal{E} - \Phi\theta^0) && (73)\\
&= \theta^0 + A^{-1}\Phi^T (\mathrm{B}^{-1} + \Phi A^{-1}\Phi^T)^{-1}(\mathcal{E} - \Phi\theta^0) && (74)
\end{aligned}$$

Now taking a sample of the posterior Gaussian process evaluated at input x to be $G = \phi(x)\zeta$ and the corresponding sample of the prior process to be $G_0 = \phi(x)\theta^0$, premultiplying the above expression by $\phi(x)$ we obtain

$$G = G_0 + \phi(x)A^{-1}\Phi^T (\mathrm{B}^{-1} + \Phi A^{-1}\Phi^T)^{-1}(\mathcal{E} - \Phi\theta^0) \tag{75}$$

which is Matheron's rule.

E RESOLVING FEATURE SCALE INDETERMINACIES IN THE NN JACOBIAN WITH THE G-PRIOR

Antorán et al. (2022c) show that for NNs with normalisation layers, the Jacobian features $\phi(\cdot) = \nabla_w f(\bar{w}, \cdot)$ corresponding to each NN layer are scaled arbitrarily. To illustrate this, we divide the NN linearisation point into the concatenation of two weight vectors $\bar{w} = [\bar{w}_0, \bar{w}_1]$. We assume the layer containing $\bar{w}_0$ is followed by a normalisation layer, but not that containing $\bar{w}_1$, which leads to the invariance

$$f([k\bar{w}_0, \bar{w}_1], \cdot) = f([\bar{w}_0, \bar{w}_1], \cdot) \tag{76}$$

for all $k > 0$.

While f is invariant to this scaling, the Jacobian feature embeddings $\phi(\cdot) = \nabla_w f(\bar{w}, \cdot)$ are not. We separate the embeddings as

$$\phi(x) = [\phi_0(\cdot), \phi_1(\cdot)] = [\nabla_{w_0} f(\bar{w}, \cdot), \nabla_{w_1} f(\bar{w}, \cdot)]. \tag{77}$$

Antorán et al. (2022c) show that, given a reference pair $([\bar{w}_0, \bar{w}_1], [\phi_0(x), \phi_1(x)])$, and for $\bar{w}_0$ normalised, scaling $\bar{w}_0$ by k results in the pair $([k\bar{w}_0, \bar{w}_1], [k^{-1}\phi_0(x), \phi_1(x)])$. Thus, using a single prior precision parameter, the regularisation strength applied to the weights multiplying $\phi_0(x)$ relative

to those multiplying $\phi_1(x)$ will increase with k. The value of k, the scale of the linearisation point, depends on exogenous factors such as learning rate or batch size—and importantly is independent of the data, since it does not affect the output.

One way to resolve this is to assign the weights $\bar{w}_0$ and $\bar{w}_1$ separate regularisation parameters and learn these using the EM procedure outlined in Section 2. However, instead, consider using the g-prior normalised features ϕ' introduced in Section 4.3, and specifically, the scaling vector corresponding to normalised and non-normalised components $s = [s_0, s_1]$. For a reference pair $([\bar{w}_0, \bar{w}_1], [s_0, s_1])$ and for $\bar{w}_0$ normalised, the k-scaled pair is $([k\bar{w}_0, \bar{w}_1]$ and

$$[\text{diag}(k^{-1}\Phi_0^T \text{B}\Phi_0 k^{-1})^{\odot -\frac{1}{2}},\ \text{diag}(\Phi_1^T \text{B}\Phi_1)^{\odot -\frac{1}{2}}] = [ks_0, s_1]$$

where $\odot$ denotes an elementwise power. Since the k-scaled features are $[k^{-1}\phi_0(\cdot), \phi_1(\cdot)]$, when applying the g-prior normalisation we recover a feature vector independent of k. This resolves the aforementioned pathology.

F A PRACTICAL IMPLEMENTATION OF SAMPLE-BASED INFERENCE AND HYPERPARAMETER LEARNING FOR LINEARISED NEURAL NETWORKS

Algorithm 1 provides a high level overview of the procedure used for our experiments. This appendix expands on this, providing fully detailed algorithms for both sampled linearised Laplace applied to classification networks and the kernelised version of the method that we use for tomographic image reconstruction.

Image classification Algorithm 2 provides an algorithm for linearised Laplace inference using the stochastic EM iteration presented in Section 3 for hyperparameter selection and the g-prior normalisation described in Section 4.3. Therein, μ denotes the softmax function. The curvature of the cross entropy loss at x_i, denoted B_i, is given by $B_i = \text{diag}(p_i) - p_i p_i^T$ for $p_i = \mu(f(\bar{w}, x_i))$ our neural network's predictive probabilities. The notation $\odot$ refers to the elementwise product and to the elementwise power when used in an exponent. We refer to the Cholesky factorisation of a positive definite matrix as its $1/2$th power.

In order to limit computational cost, we sample the stochastic regularisation terms (θ_j^n), per (7), only once at the start. Not resampling these at each E step results in the optima of the sampling objective being close for successive iterations. This comes at the cost of a small bias in our estimator which we find to be negligible in practise. We separate (θ_j^n) into a sum consisting of a prior sample from (θ_j^0) and a data dependent term, denoted (θ_j'). The former scales with $\alpha^{-1/2}$ while the latter with α^{-1} so this allows us to update each term in closed form each time α changes. We initialise our samples at (θ_j^0) at the first EM iteration. We warm-start the posterior mode $\bar{\theta}$ at the previous solution between iterations, initialising it to zero for the first iteration. We estimate the g-prior scaling vector s by noting that it relates to θ_1' as $s = \alpha^{-1}\left(\mathbb{E}[\theta_1' \odot \theta_1']\right)^{\odot -1/2}$.

We optimise both our samples ζ and posterior mean $\bar{\theta}$ using stochastic gradient descent with a Nesterov momentum parameter of 0.9. We find that Polyak averaging is very effective at reducing gradient variance when optimising the sampling objective (per Dieuleveut et al., 2017). However, it has two limitations 1) it slows down optimisation, increasing the number of steps needed 2) it doubles the memory requirement needed to store posterior samples. This decreases the number of samples that can be optimised in parallel on a single hardware accelerator. Instead we employ a linear learning rate decay schedule, which we find to work nearly as well while not increasing computational burden. The regularised classification loss $\mathcal{L}$ is non-quadratic and thus Polyak averaging is no longer optimal (Bach, 2014). Thus here we also employ a linear learning rate decay schedule. For optimising both the sampling and classification loss objectives we find that gradient clipping helps prevent oscillations at the start of training and as a result speeds up convergence.

The key hyperparameters of our algorithm are the number of samples to draw for the EM iteration, the number of EM steps to run, and SGD hyperparameters: learning rate, linear decay rate, number of steps, batch-size and gradient clipping. Empirically, we find that at most 5 EM steps are necessary for hyperparameter convergence and that as little as 3 samples can be used for the algorithm without degrading performance. Choosing SGD hyperparameters is more complicated. However, we are aided by the fact that lower loss values correspond to more precise posterior mean and sample estimates.

Algorithm 2: Sampling-based linearised Laplace inference for image classification

Inputs: Linearised network h, linearisation point $\bar{w}$, observations $x_1, \ldots, x_n$, negative log-likelihood function ℓ, initial precision $\alpha > 0$, number of samples k

Function *B(i)*:
  $p_i \leftarrow \mu(h(\bar{w}, x_i))$
  return $\operatorname{diag}(p_i) - p_i p_i^T$

for $j = 1, \ldots, k$ **do**
  $\theta_j^0 \sim \mathcal{N}(0, \alpha^{-1} I)$
  $\theta_j' \leftarrow \alpha^{-1} \sum_{i=1}^n \phi(x_i)^T \varepsilon_j$ where $\varepsilon_j \sim \mathcal{N}(0, B(i))$
  $\zeta_j \leftarrow \theta_j^0$
$\bar{\theta} \leftarrow 0$
$s \leftarrow \alpha^{-1} \left[\frac{1}{k} \sum_{j=1}^k \theta_j'^{\odot 2} \right]^{\odot -1/2}$

while α *has not converged* **do**
  for $j = 1, \ldots, k$ **do**
    $\zeta_j \leftarrow \text{SGD}_z \left(\|\Phi(s \odot z)\|_{\mathrm{B}}^2 + \alpha \|z - \theta_j^0 - (s \odot \theta_j')\|_2^2, \text{ init}=\zeta_j \right)$
  $\bar{\theta} \leftarrow \text{SGD}_\theta \left(\sum_{i=1}^n \ell(y_i, h((s \odot \theta), x_i) + \alpha \|\theta\|_2^2, \text{ init}=\bar{\theta} \right)$
  $\hat{\gamma} \leftarrow \frac{1}{k} \sum_{j=1}^k \sum_{i=1}^n \|(\zeta_j \odot s)^T \phi(x_i)^T B(i)^{\frac{1}{2}}\|_2^2$
  $\alpha' \leftarrow \hat{\gamma} / \|\bar{\theta}\|_2^2$
  for $j = 1, \ldots, k$ **do**
    $\theta_j^0 \leftarrow \sqrt{\frac{\alpha}{\alpha'}} \theta_j^0$
    $\theta_j' \leftarrow \frac{\alpha}{\alpha'} \theta_j'$
  $\alpha \leftarrow \alpha'$

Output: Optimised precision α and weight samples $\zeta_1, \ldots, \zeta_k$

Algorithm 3: Kernelised sampling-based linearised NN inference for CT reconstruction

Inputs: Linearised network h, linearisation point $\bar{w}$, measurements Y, discrete Radon transform U, U-Net Jacobian Φ, initial precision $\alpha > 0$, number of samples k, noise precision B

Function $\texttt{Kvp}$ *(v, α, UΦ, s,* B^{-1}*)*:
  return $U\Phi(\alpha^{-1} \operatorname{diag}(s^{\odot 2}))\Phi^T U^T v + \mathrm{B}^{-1} v$

$s \leftarrow (\sum_{i<m} (U_i \Phi)^{\odot 2})^{-1/2}$

while α *has not converged* **do**
  $P \leftarrow$ Compute-preconditioner($\texttt{Kvp}$)
  for $j = 1, \ldots, k$ **do**
    $\zeta_j^0 \leftarrow U\Phi(s \odot \theta_j^0) + \mathcal{E}_j$ where $\mathcal{E}_j \sim \mathcal{N}(0, \mathrm{B}^{-1})$ and $\theta_j^0 \sim \mathcal{N}(0, A^{-1})$
    $\mathtt{c}_j \leftarrow \text{CG} \left(\texttt{Kvp}, \zeta_j^0, \text{ precond.}=P \right)$
    $\zeta_j \leftarrow \zeta_j^0 - U\Phi(\alpha^{-1} \operatorname{diag}(s^{\odot 2}))\Phi^T U^T \mathtt{c}_j$
  $\delta \leftarrow U(\Phi \bar{w} - f(\bar{w}))$
  $\mathtt{c} \leftarrow \text{CG} \left(\texttt{Kvp}, Y + \delta, \text{ precond.}=P \right)$
  $\bar{\theta} \leftarrow s \odot \alpha^{-1} \Phi^T U^T \mathtt{c}$
  $\hat{\gamma} \leftarrow \frac{1}{k} \sum_{j=1}^k \|U\Phi(s \odot \zeta_j)\|_2^2$
  $\alpha' \leftarrow \hat{\gamma} / \|\bar{\theta}\|^2$
  $\alpha \leftarrow \alpha'$

Output: Optimised precision α

As a result, we can tune these parameters on the train data, no validation set is required. The specific hyperparameter values used in our experiments are provided in Appendix G.

A final thing to note is that due to the presence of normalisation layers and a dense final layer, for our classification networks, the constant-in-θ terms cancel in the linearised model and we are left with $h(\theta, x) = \phi(x)\theta$ (Antorán et al., 2022c). In our algorithm, this fact is only relevant to the computation of the posterior mode $\bar{\theta}$ as the optima of $\mathcal{L}(h(\theta, \cdot))$.

Tomographic reconstruction Algorithm 3 is the kernelised version of algorithm 2 that we use for tomographic reconstruction. This problem is described in detail in Appendix G.3.

Distinctly from the image classification setting, tomographic reconstruction is a regression problem for which we use a Gaussian likelihood with fixed noise precision $\mathrm{B} = I$. The linear model's loss function $\mathcal{L}$ is thus quadratic and the Laplace approximation is not needed. Both the sample loss and the linear model's loss can be optimised in closed form by solving a linear system of equations given by the observation covariance, i.e. the kernel matrix, $U\Phi(\alpha^{-1}\text{diag}(s^{\odot 2}))\Phi^T U^T + \mathrm{B}^{-1}$ where the linear operator U represents the discrete Radon transform and combines with the U-Net Jacobian to build the feature embedding $U\Phi$.

We solve against the kernel matrix using the preconditioned conjugate gradient (CG) method described by Gardner et al. (2018). As a preconditioner, we compute a 400-dimensional randomised eigendecomposition (alg. 5.6 in Halko et al., 2011) preconditioner, which we invert using the Woodbury identity. We find the preconditioner to provide important speedups to CG convergence and we re-estimate it after every hyperparameter update. Both computing the preconditioner and running preconditioned CG optimisation only interact with the kernel matrix by computing its products with vectors. Our algorithm defines our kernel vector product `Kvp` routine explicitly, as it is central to our implementation. We find that the GPyTorch CG implementation does not benefit from warm-starting the solution vector. Consequently, we re-draw prior and noise samples $(\theta^0, \mathcal{E})$ at every E-step.

Similarly to image classification, the key hyperparameters are the number of samples to draw for the EM iteration, the number of EM steps to run, and CG optimisation hyperparameters. Again, the number of samples can be kept low (e.g. 2) and we find around 5 steps to suffice for convergence of the prior precision α. The key conjugate gradients hyperparameters are the tolerance at which to stop optimisation and the maximum number of optimisation steps if the tolerance is not reached. We provide our choices in Appendix G.3 but note that our use of a large preconditioner results in CG always hitting the desired low error tolerance within 10 steps and never stopping due to reaching the maximum number of steps. In turn, this makes our kernelised EM algorithm notably faster than its primal form SGD-based counterpart.

A particularity of this setting is that the U-Net does not have a dense final layer. As a result, the constant-in-θ terms in the linearised function h do not cancel (see Section 4.1), leading to the appearance of the target offset term δ when solving for the posterior mean.

G Experimental details

In this appendix we provide experimental details and hyperparameter settings omitted from the main text.

G.1 MNIST experiments

MNIST m=10 way classification experiments were performed using the LeNet-style CNN architectures of increasing size employed by Antorán et al. (2022c): "LeNetSmall" (d'=14634), "LeNet" (d'=29226) and "LeNetBig" (d'=46024). We note that these models contain batch normalisation layers. Each model was trained with using SGD with momentum of 0.9 for 90 epochs with a learning rate drop of a factor of 10 every 30 epochs. The MNIST dataset was downloaded from `PyTorch torchvision`. We employ standard per-channel mean and std-dev standardisation preprocessing and two pixel shift and crop data augmentation. For posterior mode optimisation and sampling, we do not perform data augmentation as to avoid cold posterior effects (Izmailov et al., 2021). The details of our SGD approaches to convex optimisation for obtaining posterior modes and samples are as follows

- **Posterior mode optimisation**: The linearised NN weights are trained using SGD with a Nesterov momentum coefficient of 0.9, and batch size 1000 for 40 epochs. We clip gradients to a maximum norm of 1. We use an initial learning rate of $1e-2$ when using standard isotropic or layerwise Gaussian priors, and 1 for the g-prior. We employ a linear decay schedule that reduces the lr by a factor of 330 over the first 75% of the training procedure and holds it constant afterwards.
- **Sampling**: We optimise 32 samples in parallel using SGD with Nesterov momentum $(=0.9)$ and a batch size of 1000 for 20 epochs. For standard Gaussian priors (isotropic and layerwise), we use a learning rate of $2e-1$, whereas for the g-prior, we find a higher learning rate of 200 to work best.

Hyperparameter optimisation: We tuned the learning rate, decay schedule and gradient clipping strength using a rough grid search over multiple orders of magnitude. We chose the settings that reached the lowest loss values. These can be evaluated with just the train set. We chose the largest batch size that could accommodate optimising 32 samples in parallel on a single hardware accelerator. We note that posterior mode and sample optimisation converge in less than half of the total epochs we use for their optimisation. The numbers of epochs chosen were set to be large enough to ensure convergence and not tuned. A decrease in computational cost can likely be achieved by stopping sample optimisation earlier.

Baseline methods. For the comparison of learning a single precision hyperparameter and layerwise hyperparameters in Figure 3, we extend the M-step update to as $\alpha_l = {}^{\gamma_l}/_{\|\bar{\theta}_l\|_2^2}$ where l indexes each layer's attributes, as done in (Mackay, 1992; Tipping, 2001). For the MAP, diagonal covariance and KFAC covariance baselines, we use the same pre-trained models when possible (i.e. not for the ensembles or dropout baselines). Since all baselines share the same linearisation point, they also share the same mean predictions. Differences in performance among baselines are thus only due to differences in uncertainty estimation. The diagonal approximation to the covariance is constructed by first computing the diagonal of the Hessian M and the inverting it. For the KFAC covariance approximation, we exploit the equivalency between the Generalised Gauss Newton matrix (i.e. the Hessian of the linear model h) and the Fisher information matrix for exponential family likelihoods (i.e. the categorical). This allows us to formulate the Hessian as an expectation of likelihood gradients, which in turn we approximate using a single sample per training observation, as in (Daxberger et al., 2021a). For completeness, we also state the probit approximation for sampled predictive posteriors over logits. For input x and samples $\zeta_i, \ldots, \zeta_k$, the predictive probability for class $i \in \|1, \ldots, m\|$ is given by

$$\text{softmax}\left(f(\bar{w}, x) \odot (1 + \frac{\pi}{2k} \sum_{j<k} (\phi(x)\zeta_j)^{\odot 2})^{\odot -0.5}\right)_i .$$

G.2 CIFAR100 CLASSIFICATION

CIFAR100 m=100 way classification experiments were performed using ResNet18 models ($d' \approx 11M$) with specific architecture details matching the `PyTorch torchvision implementation`. We train these models using SGD with momentum of 0.9 for 300 epochs. The starting lr is 0.1 and we reduce it by a factor of 10 every 100 epochs. The CIFAR100 dataset was also downloaded using `torchvision` and our data preprocessing and augmentation also follow the default implementation from this library. For posterior mode optimisation and sampling, we do not perform data augmentation. The SGD details used to solve the convex optimisation problems required for obtaining posterior modes and drawing samples are as follows

- **Posterior mode optimisation**: The linearised NN weights are trained using SGD with Nesterov momentum $(=0.9)$ and a batch size of 2000 for 40 epochs. We employ a linear decay learning rate schedule with an initial learning rate of $1e-1$. It is decreased by a factor of 330 over the first 75% of training, and then held constant. We also employ gradient clipping with maximum norm$= 0.1$.
- **Sampling**: We optimise 6 samples in parallel using SGD with Nesterov momentum $(=0.9)$ and a batch size 100 for 20 epochs. All other details match those of posterior mode optimisation.

Upon convergence of the EM algorithm, we draw 64 further samples using the optimal prior precision by following the optimisation procedure described above. We initialise these samples at prior samples drawn with the optimised prior precision.

Hyperparameter optimisation: We tuned the learning rate, decay rate and gradient clipping strength using a rough grid search over orders of magnitude. We also chose the largest batch size that for which we could simultaneously optimise 6 samples in parallel on a single hardware accelerator. Similarly to the MNIST experiments, we did not optimise the number of optimisation epochs and instead chose large values that would ensure convergence. It is likely that our EM iteration can be sped up by decreasing the duration of the convex optimisation routines.

Details for baselines and hyperparameters not mentioned explicitly in this subsection match those given for MNIST in the previous subsection.

G.2.1 EFFICIENT κ-ADIC SAMPLING

Osband et al. (2022; 2021) introduced *dyadic* test input sampling ($\kappa = 2$) as a practical way of evaluating joint predictions in discriminative tasks. This method samples $\kappa = 2$ random anchor points from the test dataset, and then randomly resamples them to create a batch of size $\tau = 10$. Test log-likelihood is evaluated jointly for each batch as

$$\log \int \exp\left(\sum_{i\leq\tau} \ell(y_i, f(\theta, x_i))\right) d\Pi,$$

for f the model being evaluated and Π its posterior distribution over model parameters. This quantity can be estimated with posterior samples $\zeta_1, \ldots, \zeta_k \sim \Pi$ as

$$\log \frac{1}{k} \sum_{j\leq k} \exp\left(\sum_{i\leq\tau} \ell(y_i, f(\zeta_j, x_i))\right).$$

We extend this evaluation approach to larger κ and τ values without increasing computational cost. We randomly sample κ integers $\{b_1, \ldots, b_\kappa\}$ such that they sum to τ, i.e $\sum_i^\kappa k_i = \tau$. The joint log-likelihood over the batch of size τ with κ unique datapoints can then be estimated as

$$\log \frac{1}{k} \sum_{j\leq k} \exp\left(\sum_{l\leq\kappa} b_l \ell(y_l, f(\zeta_j, x_l))\right).$$

where the inner sum is over the κ distinct elements in the batch instead of the "total batch size" τ. This is equivalent to the formulation proposed in Osband et al. (2022) for dyadic sampling, when $\kappa = 2$ and $\tau = 10$. We note that it is not possible to achieve *augmented dyadic* sampling, as described in (Osband et al., 2021), with this approach. However the authors mention that there is not a significant difference in the relative performance of methods when using augmented dyadic sampling compared to regular dyadic sampling. We introduce a final step however, which is to repeat the computation for multiple shuffles (10) of the test dataset. This eliminates variance in our results from the choice of the κ observations which get grouped together in each batch.

G.3 TOMOGRAPHIC RECONSTRUCTION

Problem setup Tomographic reconstruction consists in solving a linear inverse problem in imaging where we observe a set of measurements $y \in \mathbb{R}^m$, which we assume to be generated as $y = Ux^* + \eta$ for $U \in \mathbb{R}^{m\times d}$ the discrete Radon transform, $x^* \in \mathbb{R}^d$ the image to reconstruct and $\eta \sim \mathcal{N}(0, I)$ random noise. We have $m \ll d$, making the problem underconstrained. Our specific tomographic reconstruction task closely follows the one from Barbano et al. (2021). We perform a sparse-view reconstruction of an image of a slice of a walnut from a sub-sampled set of measurements. Specifically, from the full measurement set of (Der Sarkissian et al., 2019a), which containing scans at 1200 equidistant angles over $[0, 360^\circ)$, we choose our measurement set by subsampling angles by a factor of either 10x or 20x, leading to measurements of size $m = 15360$ or $m = 7680$. As in Barbano et al. (2021); Antorán et al. (2022b), we reduce the original 3D scan geometry to the 2D slice of interest by selecting the relevant subset of measurement pixels. We assemble the Radon operator U as a sparse matrix taking in an image of resolution $(501\text{px})^2$ and outputting a measurement tensor coherent with the described geometry.

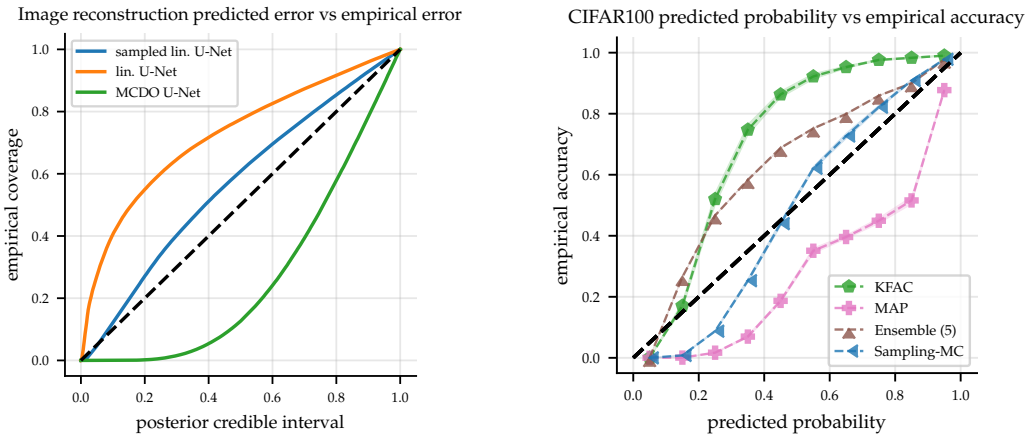

Figure 8: Left: empirical coverage of test targets for posterior credible intervals of increasing width for our U-net tomographic reconstruction experiment (Section 5.3). Right: confidence vs accuracy plot (also known as a reliability diagram) for our CIFAR100 classification experiment (Section 5.2).

Methods To provide a reconstruction, we use the Deep Image prior (Ulyanov et al., 2020) which trains the parameters $w \in \mathbb{R}^{d'}$ of a fully convolutional U-Net autoencoder $f : \mathbb{R}^{d'} \to \mathbb{R}^{d}$, where the input is fixed and thus absent from our notation, until a satisfactory reconstruction $f(\bar{w})$ is obtained. The U-Net network architecture is the one proposed in (Baguer et al., 2020). The optimisation of the U-Net parameters follows Barbano et al. (2021). To estimate the uncertainty in this reconstruction, we linearise the U-Net around $\bar{w}$, as described in Section 4.1. This leaves us with a model affine in the parameters and with design matrix $U\Phi \in \mathbb{R}^{m \times d'}$. We may now proceed with linear model inference. While (Antorán et al., 2022b) use the traditional EM iteration described in Section 2, we use the sample-based one from Section 3. For evaluation, we use the non-sparse reconstruction (using 1200 angles) provided by (Der Sarkissian et al., 2019a) as the ground truth image x^*. To evaluate joint log-likelihoods we estimate the predictive covariance matrix for patches of neighbouring pixels using samples. Covariance matrix estimates from samples are known to be unreliable. We use the stabilised formulation of (Maddox et al., 2019): $\hat{\Sigma} = \frac{1}{2k}\left[\sum_{j=1}^{k} \hat{x}_j^2 + \hat{x}_j \hat{x}_j^T\right]$ for $(\hat{x}_j)_{j=1}^{k}$ samples from the predictive posterior over a patch. We then construct predictive distributions over pixels as $\mathcal{N}(f(\bar{w}), \hat{\Sigma})$.

Hyperparameters We employ a low CG tolerance of $1e-3$ and a maximum number of iterations of 150, which is never reached in practise as the error tolerance level is always hit in less steps.

H CALIBRATION OF PREDICTIVE DISTRIBUTIONS

This appendix evaluates the calibration of the predictive distributions provided by the methods under consideration in our CIFAR100 classification experiment (Section 5.2) and U-net image reconstruction experiment (Section 5.3).

For classification, we separate our predicted probabilities into 10 equal width bins between 0 and 1. For each bin, we plot the proportion of targets that coincide with the class for which the predicted probability falls into the bin. This is shown on the right hand side of Figure 8. Consistent with the results from the main text, KFAC overestimates uncertainty at all confidence levels whereas MAP underestimates it. Both sample-based linearised Laplace and ensembling show significantly improved calibration. While ensembles show a small amount of uncertainty overestimation consistently, our method underestimates uncertainty for low predicted probabilities and overestimates it for large predicted probabilities.

For image reconstruction regression, we first compute normalised residuals by subtracting our predictions from the targets and dividing by the predictive standard deviation. Our predictive distribution for these normalised residuals is the centered unit variance Gaussian. We consider posterior credible intervals centered at 0 and of increasing width and plot the proportion of test points

Table 3

	κ	MAP	Ensemble (5)	KFAC	Sampling	KFAC-LL *	subnetwork *
marginal LL	1	-1.40 $\pm$ 0.00	**-0.90 $\pm$ 0.00**	-1.12 $\pm$ 0.01	-1.07 $\pm$ 0.01	-1.06 $\pm$ 0.01	-1.21 $\pm$ 0.01
joint LL	2	-13.97 $\pm$ 0.01	-6.86 $\pm$ 0.01	**-4.92 $\pm$ 0.04**	-5.14 $\pm$ 0.04	-5.41 $\pm$ 0.05	-8.38 $\pm$ 0.07
	3	-27.89 $\pm$ 0.03	-14.17 $\pm$ 0.03	-10.83 $\pm$ 0.12	**-10.77 $\pm$ 0.09**	-11.15 $\pm$ 0.12	-16.59 $\pm$ 0.13
	4	-41.83 $\pm$ 0.03	-22.29 $\pm$ 0.04	-19.02 $\pm$ 0.22	**-18.04 $\pm$ 0.18**	-18.21 $\pm$ 0.18	-25.47 $\pm$ 0.18
	5	-55.89 $\pm$ 0.02	-31.07 $\pm$ 0.09	-29.40 $\pm$ 0.40	**-26.75 $\pm$ 0.26**	**-26.50 $\pm$ 0.26**	-34.91 $\pm$ 0.30

that fall within them in the left side plot of Figure 8. We find dropout inference to underestimate the magnitude of the residuals across all credible interval widths. Linearised inference with a single EM step, as in (Antorán et al., 2022b), consistently overestimates uncertainty. Our approach, which performs 5 steps of EM, overestimates uncertainty, but to a much smaller degree, presenting the best overall calibration. The latter two approaches consist of the same model but with different regularisation strength. The difference between the two reveals the paramount importance of tuning the prior precision hyperparameter well.

I Additional Experiments

I.1 CIFAR100 Classification

Additional Baselines for marginal LL. In the main text, we report the predictive uncertainty of our method versus point-estimated NNs (MAP), 5 ensembles of point-estimate NNs (Ensemble 5), and a KFAC estimate (KFAC). Here, we report further comparisons with other baselines standard-in-literature: a diagonal approximation of the Laplace covariance matrix (diag), a Laplace approximation over a selected subnetwork of the original NN (subnetwork*), and a Laplace approximation over the last-layer of the NN further approximated by a KFAC matrix (KFAC-LL*). We distinguish the last two methods in a separate category (*), since these methods require cross-validation with a held-out set in order to tune hyperparameters. For these methods, we use 50 held-out points from the test set, and evaluate on the remaining 9950 points. Figure 9 shows that somewhat surprisingly, KFAC-LL performs quite competitively with our approach, and even with deep ensembles. However, it is constrained to require a cross-validation set in order to tune hyperparameters.

Predictive uncertainty vs number of samples. In the main text, we report predictive performance for our method with 64 samples. However, it is possible that drawing many samples can be computationally expensive. We compare the quality of predictive uncertainty obtained on the in-distribution test set and 5 corruptions vs the number of samples used for estimating the predictive variance, and results are shown in Figure 10.

Additional baselines for joint LL. Similar to the marginal predictions, we also report KFAC-LL and subnetwork inference as two additional baselines for κ-adic sampling in Table 3. KFAC-LL is once again quite competitive with our approach, but requires a held-out validation set.

I.2 Tomographic Reconstruction

Measuring prediction calibration Figure 11 and Figure 12 are complementary to Figure 6 and Figure 7, and report EM iteration and performance for m=15360. In Figure 13 and Figure 14, we show additional insights on the calibration of the uncertainty estimates obtained via the proposed method when compared to the MCDO approach.

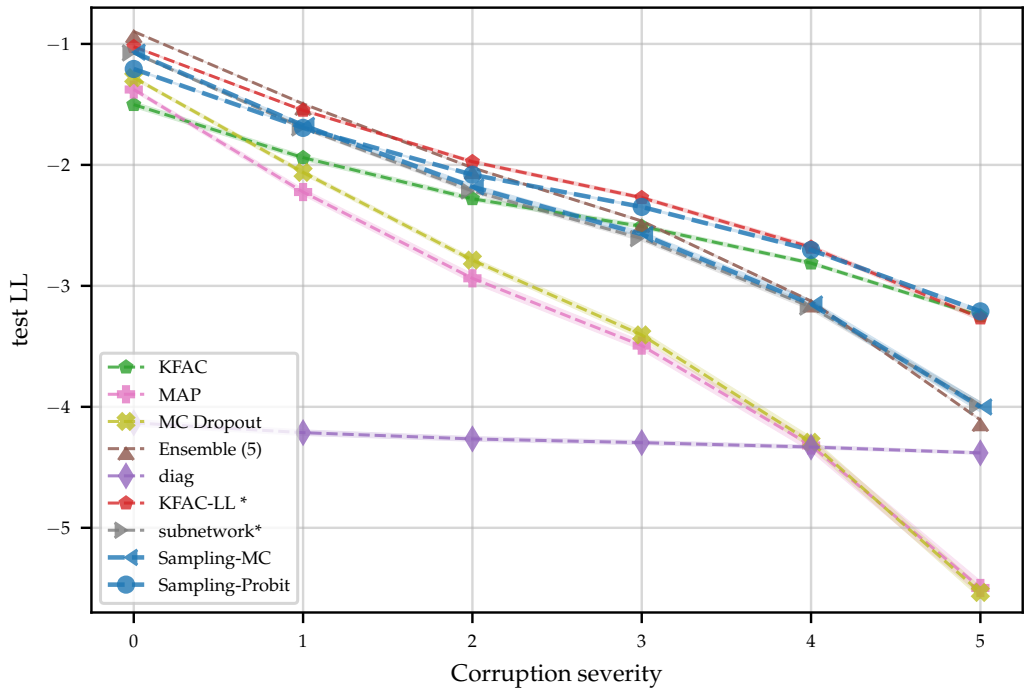

Figure 9: Performance under distribution shift for ResNet18 and CIFAR100. We add additional baselines to Figure 5: MC Dropout and diag-Laplace. We also add KFAC-LL and subnetwork inference, methods that require a held-out validation set to tune hyperparameters. Finally we report Sampling-MC, the predictive uncertainty obtained by our method using an MC estimate of the predictive distribution.

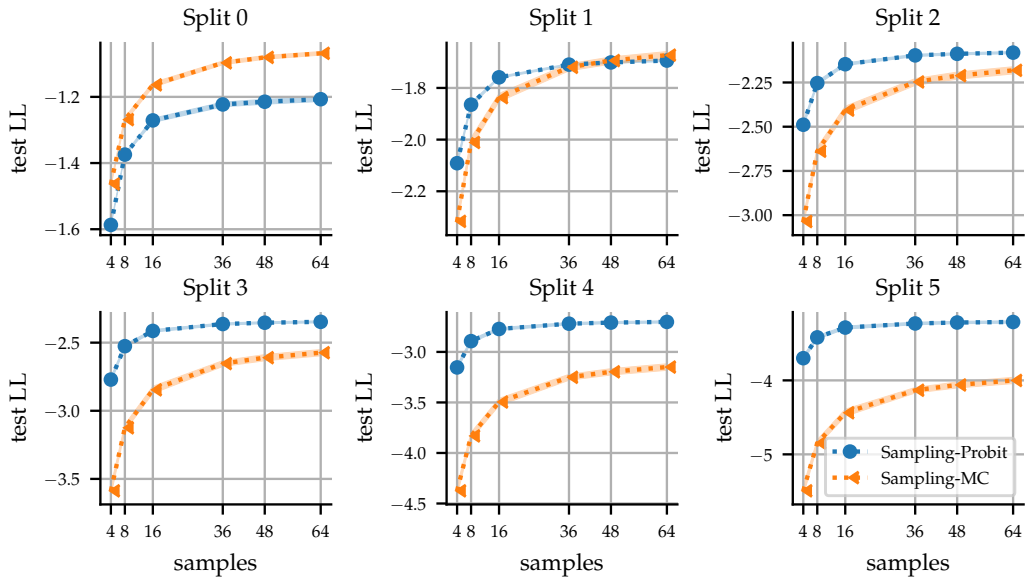

Figure 10: Predictive Performance for CIFAR100 vs the number of samples used. The Probit approximation tends to require fewer samples to estimate the marginal variance, but is more underconfident under distribution shift. On the other hand, estimating the predictive distribution using an MC estimate (Sampling-MC) results in better performance in-distribution, however it requires more samples in order to predict uncertainty under distribution shift.

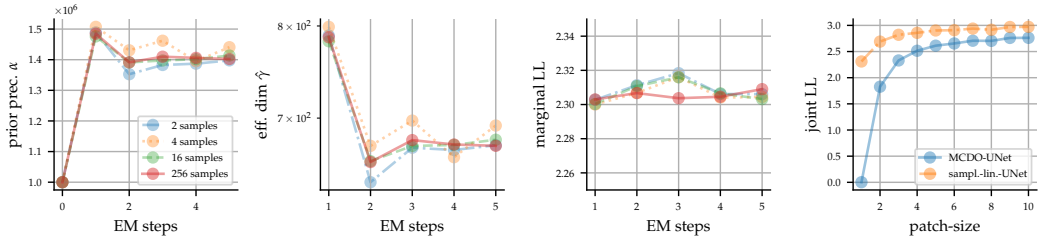

Figure 11: Sample-based EM iteration convergence for tomographic reconstruction given $m = 15360$. The prior precision α, the effective dimension $\hat{\gamma}$, and the marginal LL as a function of the EM step along showing marginal and joint LL for different patch-size.

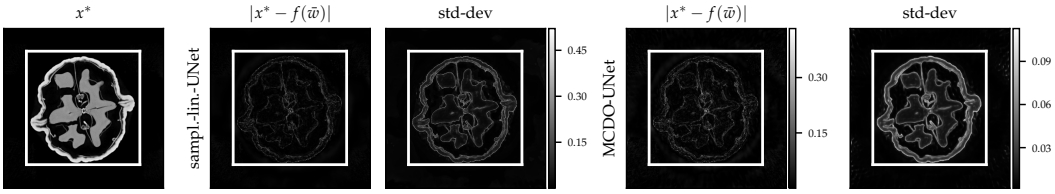

Figure 12: Absolute error computed between the reconstructed Walnut given $m = 15360$ and the ground-truth for both lin.-DIP and MCDO-DIP along showing respective uncertainty estimates.

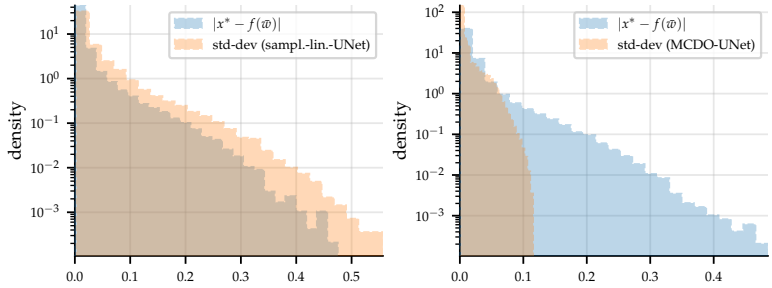

Figure 13: Histogram of the absolute error computed between the reconstructed Walnut given $m = 7680$ and the ground-truth for both lin.-DIP and MCDO-DIP, and of the respective standard deviations.

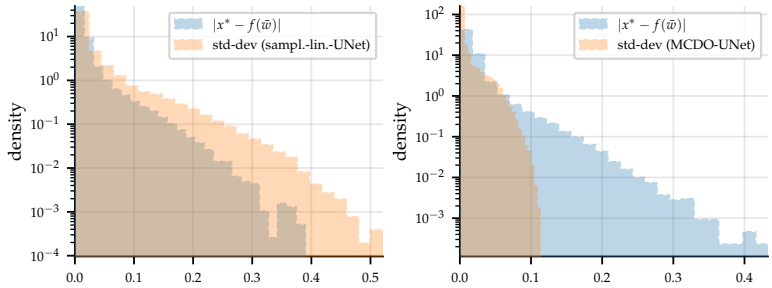

Figure 14: Histogram of the absolute error computed between the reconstructed Walnut given $m = 15360$ and the ground-truth for both lin.-DIP and MCDO-DIP, and of the respective standard deviations.