A Derivations of the augmented (pseudo) prior

A.1 Inducing auxiliary variables: multivariate Gaussian case

Suppose each weight matrix has an isotropic Gaussian prior with zero mean, i.e. $\operatorname{vec}(W) \sim \mathcal{N}(0, \sigma^2 I)$ where vec concatenates the columns of a matrix into a vector and σ is the standard deviation. Augmenting this Gaussian with an auxiliary variable U that also has a mean of zero and some covariance that we are free to parameterise, the joint distribution is

$$\begin{pmatrix} \operatorname{vec}(W) \\ \operatorname{vec}(U) \end{pmatrix} \sim \mathcal{N}(0, \Sigma) \quad \text{with} \quad L = \begin{pmatrix} \sigma I & 0 \\ Z & D \end{pmatrix} \quad \text{s.t.} \quad \Sigma = LL^\top = \begin{pmatrix} \sigma^2 I & \sigma Z^\top \\ \sigma Z & ZZ^\top + D^2 \end{pmatrix}$$

where D is a positive diagonal matrix and Z a matrix with arbitrary entries. Through defining the Cholesky decomposition of Σ we ensure its positive definiteness. By the usual rules of Gaussian marginalisation, the augmented model leaves the marginal prior on W unchanged. Further, we can analytically derive the conditional distribution on the weights given the inducing weights:

$$p(\operatorname{vec}(W) | \operatorname{vec}(U)) = \mathcal{N}(\mu_{W|U}, \Sigma_{W|U}), \tag{14}$$
$$\mu_{W|U} = \sigma Z^\top \Psi^{-1} \operatorname{vec}(U), \quad \Sigma_{W|U} = \sigma^2 (I - Z^\top \Psi^{-1} Z), \quad \Psi = ZZ^\top + D^2.$$

For inference, we now need to define an approximate posterior over the joint space $q(W, U)$. We will do so by factorising it as $q(W, U) = q(W|U)q(U)$. Factorising the prior in the same way leads to the following KL term in the ELBO:

$$\mathbb{KL}\left[q(W, U) || p(W, U)\right] = \mathbb{E}_{q(U)}\left[\mathbb{KL}\left[q(W|U) || p(W|U]\right] + \mathbb{KL}\left[q(U) || p(U)\right]$$

A.2 Inducing auxiliary variables: matrix normal case

Now we introduce the inducing variables in matrix space, and, in addition to the inducing weight U, we pad in two inducing matrices U_r, U_c, such that the full augmented prior is:

$$\begin{pmatrix} W & U_c \\ U_r & U \end{pmatrix} \sim p(W, U_c, U_r, U) := \mathcal{MN}(0, \Sigma_r, \Sigma_c), \tag{15}$$
$$\text{with} \quad L_r = \begin{pmatrix} \sigma_r I & 0 \\ Z_r & D_r \end{pmatrix} \quad \text{s.t.} \quad \Sigma_r = L_r L_r^\top = \begin{pmatrix} \sigma_r^2 I & \sigma_r Z_r^\top \\ \sigma_r Z_r & Z_r Z_r^\top + D_r^2 \end{pmatrix},$$
$$\text{and} \quad L_c = \begin{pmatrix} \sigma_c I & 0 \\ Z_c & D_c \end{pmatrix} \quad \text{s.t.} \quad \Sigma_c = L_c L_c^\top = \begin{pmatrix} \sigma_c^2 I & \sigma_c Z_c^\top \\ \sigma_c Z_c & Z_c Z_c^\top + D_c^2 \end{pmatrix}.$$

Matrix normal distributions have similar marginalisation and conditioning properties as multivariate Gaussians. As such, the marginal both over some set of rows and some set of columns is still a matrix normal. Hence, $p(W) = \mathcal{MN}(0, \sigma_r^2 I, \sigma_c^2 I)$, and by choosing $\sigma_r \sigma_c = \sigma$ this matrix normal distribution is equivalent to the multivariate normal $p(\operatorname{vec}(W)) = \mathcal{N}(0, \sigma^2 I)$. Also $p(U) = \mathcal{MN}(0, \Psi_r, \Psi_c)$, where again $\Psi_r = Z_r Z_r^\top + D_r^2$ and $\Psi_c = Z_c Z_c^\top + D_c^2$. Similarly, the conditionals on some rows or columns are matrix normal distributed:

$$U_c | U \sim \mathcal{MN}(\sigma_r Z_r^\top \Psi_r^{-1} U, \sigma_r^2 (I - Z_r^\top \Psi_r^{-1} Z_r), \Psi_c), \tag{16}$$
$$U_r | U \sim \mathcal{MN}(U \Psi_c^{-1} \sigma_c Z_c, \Psi_r, \sigma_c^2 (I - Z_c^\top \Psi_c^{-1} Z_c)), \tag{17}$$
$$W | U_c \sim \mathcal{MN}\left(U_c \Psi_c^{-1} \sigma_c Z_c, \sigma_r^2 I, \sigma_c^2 (I - Z_c^\top \Psi_c^{-1} Z_c)\right) \tag{18}$$
$$W, U_r | U_c, U \sim \mathcal{MN}(\begin{pmatrix} U_c \\ U \end{pmatrix} \Psi_c^{-1} \sigma_c Z_c, \Sigma_r, \sigma_c^2 (I - Z_c^\top \Psi_c^{-1} Z_c)), \tag{19}$$
$$W | U_r, U_c, U \sim \mathcal{MN}(M_W, \sigma_r^2 (I - Z_r^\top \Psi^{-1} Z_r), \sigma_c^2 (I - Z_c^\top \Psi_c^{-1} Z_c)), \tag{20}$$
$$M_W = \sigma (Z_r^\top \Psi_r^{-1} U_r + U_c \Psi_c^{-1} Z_c - Z_r^\top \Psi_r^{-1} U \Psi_c^{-1} Z_c). \tag{21}$$

B KL divergence for rescaled conditional weight distributions

For the conditional distribution on the weights, in the simplest case we set $q(W|U) = p(W|U)$, hence the KL divergence would be zero. For the most general case of arbitrary Gaussian distributions with $q = \mathcal{N}(\mu_q, \Sigma_q)$ and $p = \mathcal{N}(\mu_p, \Sigma_p)$, the KL divergence is:

$$\mathbb{KL}\left[q||p\right] = \frac{1}{2}(\log \frac{\det \Sigma_p}{\det \Sigma_q} - d + \mathrm{tr}(\Sigma_p^{-1}\Sigma_q) + (\mu_p - \mu_q)^\top \Sigma_p^{-1}(\mu_p - \mu_q)),$$

where d is the number of elements of μ. As motivated, it is desirable to make $q(W|U)$ similar to $p(W|U)$. We consider a scalar rescaling of the covariance, i.e. for $p = \mathcal{N}(\mu, \Sigma)$ we set $q = \mathcal{N}(\mu, \lambda^2\Sigma)$. This leads to the final term, which is the Mahalanobis distance between the means under p, cancelling out entirely and the log determinant and trace terms becoming a function of λ only: with $d = \dim(\mathrm{vec}(W))$,

$$\begin{aligned}\mathbb{KL}\left[q||p\right] &= \frac{1}{2}(\log \frac{\det \Sigma}{\det \lambda^2\Sigma} - d + \mathrm{tr}(\Sigma^{-1}\lambda^2\Sigma)) \\ &= \frac{1}{2}(\log \frac{\det \Sigma}{\lambda^{2d} \det \Sigma} - d + \mathrm{tr}(\lambda^2 I)) \\ &= \frac{1}{2}(-2d \log \lambda - d + d\lambda^2) \\ &= d(\frac{1}{2}\lambda^2 - \log \lambda - \frac{1}{2}).\end{aligned}$$

C The extended Matheron's rule to matrix normal distributions

The original Matheron's rule (Journel & Huijbregts, 1978; Hoffman & Ribak, 1991; Doucet, 2010) for sampling conditional Gaussian variables states the following. If the joint multivariate Gaussian distribution is

$$\begin{pmatrix}\mathrm{vec}(W) \\ \mathrm{vec}(U)\end{pmatrix} \sim p(\mathrm{vec}(W), \mathrm{vec}(U)) := \mathcal{N}(\mathbf{0}, \Sigma), \quad \Sigma = \begin{pmatrix}\Sigma_{WW} & \Sigma_{WU} \\ \Sigma_{UW} & \Sigma_{UU}\end{pmatrix},$$

then, conditioned on U, sampling $W \sim p(\mathrm{vec}(W), \mathrm{vec}(U))$ can be done as

$$\mathrm{vec}(W) = \mathrm{vec}(\bar{W}) + \Sigma_{WU}\Sigma_{UU}^{-1}(\mathrm{vec}(U) - \mathrm{vec}(\bar{U})), \quad \mathrm{vec}(\bar{W}), \mathrm{vec}(\bar{U}) \sim \mathcal{N}(\mathbf{0}, \Sigma).$$

Matheron's rule can provide significant speed-ups if $\mathrm{vec}(U)$ has significantly smaller dimensions than that of $\mathrm{vec}(W)$, and the Cholesky decomposition of Σ can be computed with low costs (e.g. due to the specific structure in Σ). Recall from the main text that the augmented prior is

$$p(\mathrm{vec}(W), \mathrm{vec}(U)) = \mathcal{N}\left(0, \begin{pmatrix}\sigma_c^2 I \otimes \sigma_r^2 I & \sigma_c Z_c^\top \otimes \sigma_r Z_r^\top \\ \sigma_c Z_c \otimes \sigma_r Z_r & \Psi_c \otimes \Psi_r\end{pmatrix}\right),$$

and the corresponding conditional distribution is:

$$p(\mathrm{vec}(W)|\,\mathrm{vec}(U)) = \mathcal{N}(\sigma_c\sigma_r \,\mathrm{vec}(Z_r\Psi_r^{-1}U\Psi_c^{-1}Z_c^\top), \sigma_c^2\sigma_r^2(I - Z_c^\top\Psi_c^{-1}Z_c \otimes Z_r^\top\Psi_r^{-1}Z_r)). \quad (22)$$

Therefore, while $\dim(\mathrm{vec}(U))$ is indeed significantly smaller than of $\dim(\mathrm{vec}(W))$ by construction, the joint covariance matrix does not support fast Cholesky decompositions, meaning that Matheron's rule for efficient sampling does not directly apply here.

However, in the full augmented space, the joint distribution does have an efficient matrix normal form: $p(W, U_c, U_r, U_c) = \mathcal{MN}(0, \Sigma_r, \Sigma_c)$. Furthermore, the row and column covariance matrices Σ_r and Σ_c are parameterised by their Cholesky decompositions, meaning that sampling from the joint distribution $p(W, U_c, U_r, U)$ can be done in a fast way. Importantly, Cholesky decompositions for $p(U)$'s row and column covariance matrices Ψ_r and Ψ_c can be computed in $\mathcal{O}(M_{out}^3)$ and $\mathcal{O}(M_{in}^3)$ time, respectively, which are much faster than the multi-variate Gaussian case that requires $\mathcal{O}(M_{in}^3 M_{out}^3)$ time. Observing these, we extend Matheron's rule to sample $p(W|U)$ where $p(W, U)$ is the marginal distribution of $p(W, U_c, U_r, U_c) = \mathcal{MN}(0, \Sigma_r, \Sigma_c)$.

In detail, for drawing a sample from $p(W|U)$ we need to draw a sample from the joint $p(W, U)$. To do so, we sample from the augmented prior $\bar{W}, \bar{U}_c, \bar{U}_r, \bar{U} \sim p(\bar{W}, \bar{U}_c, \bar{U}_r, \bar{U}) = \mathcal{MN}(0, \Sigma_r, \Sigma_c)$, computed using the Cholesky decompositions of Σ_r and Σ_c:

$$\begin{pmatrix}\bar{W} & \bar{U}_c \\ \bar{U}_r & \bar{U}\end{pmatrix} = \begin{pmatrix}\sigma_r I & 0 \\ Z_r & D_r\end{pmatrix}\begin{pmatrix}E_1 & E_2 \\ E_3 & E_4\end{pmatrix}\begin{pmatrix}\sigma_c I & Z_c^\top \\ 0 & D_c\end{pmatrix},$$

where $E_1 \in \mathbb{R}^{d_{out} \times d_{in}}$, $E_2 \in \mathbb{R}^{d_{out} \times M_{in}}$, $E_3 \in \mathbb{R}^{M_{out} \times d_{in}}$, $E_4 \in \mathbb{R}^{M_{out} \times M_{in}}$ are standard Gaussian noise samples, and $\bar{W} \in \mathbb{R}^{d_{out} \times d_{in}}$ and $\bar{U} \in \mathbb{R}^{M_{out} \times M_{in}}$. Then we construct the conditional sample $W \sim p(W|U)$ as follows, similar to Matheron's rule in the multivariate Gaussian case:

$$W = \bar{W} + \sigma_r \sigma_c Z_r^\top \Psi_r^{-1} (U - \bar{U}) \Psi_c^{-1} Z_c. \tag{23}$$

From the above equations we see that $\bar{U}_r$ and $\bar{U}_c$ do not contribute to the final W sample. Therefore we do not need to compute $\bar{U}_r$ and $\bar{U}_c$, and we write the separate expressions for $\bar{W}$ and $\bar{U}$ as:

$$\bar{W} = \sigma_r \sigma_c E_1, \quad \bar{U} = \underbrace{Z_r E_1 Z_c^\top}_{\bar{U}_1} + \underbrace{Z_r E_2 D_c}_{\bar{U}_2} + \underbrace{D_r E_3 Z_c^\top}_{\bar{U}_3} + \underbrace{D_r E_4 D_c}_{\bar{U}_4}. \tag{24}$$

Note that $\bar{U}$ is a sum of four samples from matrix normal distributions. In particular, we have that:

$$\bar{U}_2 \overset{d}{\sim} \mathcal{MN}(0, Z_r Z_r^\top, D_c^2) \quad \text{and} \quad \bar{U}_3 \overset{d}{\sim} \mathcal{MN}(0, D_r^2, Z_c Z_c^\top).$$

Hence instead of sampling the "long and thin" Gaussian noise matrices E_2 and E_3, we can reduce variance by sampling standard Gaussian noise matrices $\tilde{E}_2, \tilde{E}_3 \in \mathbb{R}^{M_{out} \times M_{in}}$, and calculate $\bar{U}$ as

$$\bar{U} = Z_r E_1 Z_c^\top + \hat{L}_r \tilde{E}_2 D_c + D_r \tilde{E}_3 \hat{L}_c^\top + D_r E_4 D_c. \tag{25}$$

This is enabled by calculating the Cholesky decompositions $\hat{L}_r \hat{L}_r^\top = Z_r Z_r^\top$ and $\hat{L}_c \hat{L}_c^\top = Z_c Z_c^\top$, which have $\mathcal{O}(M_{out}^3)$ and $\mathcal{O}(M_{in}^3)$ run-time costs, respectively. As a reminder, the Cholesky factors are square matrices, i.e. $\hat{L}_r \in \mathbb{R}^{M_{out} \times M_{out}}$, $\hat{L}_c \in \mathbb{R}^{M_{in} \times M_{in}}$). We name the approach the extended Matheron's rule for sampling conditional Gaussians when the full joint has a matrix normal form.

As to verify the proposed approach, we compute the mean and the variance of the random variable W defined in (23), and check if they match the mean and variance of (22). First as $\bar{W}, \bar{U}$ have zero mean, it is straightforward to verify that $\mathbb{E}[W] = \sigma_r \sigma_c Z_r^\top \Psi_r^{-1} U \Psi_c^{-1} Z_c$ which matches the mean of (22). For the variacne of $\text{vec}(W)$, it requires computing the following terms:

$$\begin{aligned} \mathbb{V}(\text{vec}(W)) = & \mathbb{V}(\text{vec}(\bar{W})) + \mathbb{V}(\text{vec}(\sigma_r \sigma_c Z_r^\top \Psi_r^{-1} \bar{U} \Psi_c^{-1} Z_c)) \\ & - 2\text{Cov}[\text{vec}(W), \text{vec}(\sigma_r \sigma_c Z_r^\top \Psi_r^{-1} \bar{U} \Psi_c^{-1} Z_c)] := A_1 + A_2 - 2A_3. \end{aligned} \tag{26}$$

First it can be shown that

$$\begin{aligned} A_1 &= \sigma_r^2 \sigma_c^2 I \quad \text{since } \bar{W} \sim \mathcal{MN}(0, \sigma_r^2 I, \sigma_c^2 I), \\ A_2 &= \sigma_r^2 \sigma_c^2 Z_c^\top \Psi_c^{-1} Z_c \otimes Z_r^\top \Psi_r^{-1} Z_r \quad \text{since } Z_r^\top \Psi_r^{-1} \bar{U} \Psi_c^{-1} Z_c \overset{d}{\sim} \mathcal{MN}(0, Z_r^\top \Psi_r^{-1} Z_r, Z_c^\top \Psi_c^{-1} Z_c). \end{aligned}$$

For the correlation term A_3, we notice that $\bar{W}$ and $\bar{U}$ only share the noise matrix E_1 in the joint sampling procedure eq. (24). This also means

$$A_3 = \sigma_r^2 \sigma_c^2 \text{Cov}[\text{vec}(E_1), \text{vec}(Z_r^\top \Psi_r^{-1} Z_r E_1 Z_c^\top \Psi_c^{-1} Z_c)] = \sigma_r^2 \sigma_c^2 Z_c^\top \Psi_c^{-1} Z_c \otimes Z_r^\top \Psi_r^{-1} Z_r.$$

Plugging in A_1, A_2, A_3 into eq. (26) verifies that $\mathbb{V}(\text{vec}(W))$ matches the variance of the conditional distribution $p(\text{vec}(W)|\,\text{vec}(U))$, showing that the proposed extended Matheron's rule indeed draws samples from the conditional distribution.

As for sampling W from $q(W|U)$, since it has the same mean but a rescaled covariance as compared with $p(W|U)$, we can compute the samples by adapting the extend Matheron's rule as follwos. Notice that the mean of W in (23) is $\mathbb{E}[W|U] = \sigma_r \sigma_c Z_r^\top \Psi_r^{-1} U \Psi_c^{-1} Z_c$, therefore by rearranging terms, eq. (23) can be re-written as

$$W = Z_r^\top \Psi_r^{-1} U \Psi_c^{-1} Z_c + [\bar{W} - \sigma_r \sigma_c Z_r^\top \Psi_r^{-1} \bar{U} \Psi_c^{-1} Z_c] := \text{mean} + \text{noise}.$$

So sampling from $q(W|U)$ can be done by rescaling the noise term in the above equation with the scale parameter λ. In summary, the extended Matheron's rule for sampling $q(W|U)$ is as follows:

$$W = \lambda \bar{W} + \sigma_r \sigma_c Z_r^\top \Psi_r^{-1} (U - \lambda \bar{U}) \Psi_c^{-1} Z_c, \quad \bar{W}, \bar{U} \sim p(\bar{W}, \bar{U}_c, \bar{U}_r, \bar{U}). \tag{27}$$

Plugging in $\sigma_r \sigma_c = \sigma$ here returns the conditional sampling rule (10) in the main text.

D FUNCTION-SPACE VIEW OF INDUCING WEIGHTS

Here we present the detailed derivations of Section 3.3. Assume a neural network layer with weight W computes the following transformation of the input $\mathbf{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N], \boldsymbol{x}_i \in \mathbb{R}^{d_{in} \times 1}$:

$$\mathbf{F} = W\mathbf{X}, \ \mathbf{H} = g(\mathbf{F}), \quad W \in \mathbb{R}^{d_{out} \times d_{in}}, \mathbf{X} \in \mathbb{R}^{d_{in} \times N}, g(\cdot) \text{ is the non-linearity.}$$

Therefore the Gaussian prior $p(W) = \mathcal{N}(0, \sigma^2 I)$ induces a Gaussian distribution on the linear transformation output $\mathbf{F}$, in fact each of the rows in $\mathbf{F} = [\mathbf{f}_1, ..., \mathbf{f}_{d_{out}}]^\top, \mathbf{f}_i \in \mathbb{R}^{N \times 1}$ has a Gaussian process form with linear kernel:

$$\mathbf{f}_i | \mathbf{X} \sim \mathcal{GP}(\mathbf{0}, \mathbf{K}_{\mathbf{XX}}), \quad \mathbf{K}_{\mathbf{XX}}(m, n) = \sigma^2 \boldsymbol{x}_m^\top \boldsymbol{x}_n. \tag{28}$$

Performing inference on $\mathbf{F}$ directly has $\mathcal{O}(N^3 + d_{out}N^2)$ cost, so a sparse approximation is needed. Slightly different from the usual approach, we introduce "scaled noisy inducing outputs" $U_c = [\mathbf{u}_1^c, ..., \mathbf{u}_{d_{out}}^c]^\top \in \mathbb{R}^{d_{out} \times M_{in}}$ in the following way, using shared inducing inputs $Z_c^\top \in \mathbb{R}^{d_{in} \times M_{in}}$:

$$p(\mathbf{f}_i, \hat{\mathbf{u}}_i | \mathbf{X}) = \mathcal{GP}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K}_{\mathbf{XX}} & \mathbf{K}_{\mathbf{X}Z_c} \\ \mathbf{K}_{Z_c\mathbf{X}} & \mathbf{K}_{Z_cZ_c} \end{pmatrix}\right), \quad p(\mathbf{u}_i^c | \hat{\mathbf{u}}_i) = \mathcal{N}\left(\frac{\hat{\mathbf{u}}_i}{\sigma_c}, \sigma_r^2 D_c^2\right),$$

with $\mathbf{K}_{Z_c\mathbf{X}} = \sigma^2 Z_c \mathbf{X}$ and $\mathbf{K}_{Z_cZ_c} = \sigma^2 Z_c Z_c^\top$. By marginalising out the "noiseless inducing outputs" $\hat{\mathbf{u}}_i$, we have the joint distribution $p(\mathbf{f}_i, \mathbf{u}_i)$ as

$$p(\mathbf{u}_i^c) = \mathcal{N}(\mathbf{0}, \sigma_r^2 \Psi_c), \ \Psi_c = Z_c Z_c^\top + D_c^2,$$
$$p(\mathbf{f}_i | \mathbf{X}, \mathbf{u}_i^c) = \mathcal{N}\left(\sigma_c \sigma^{-2} \mathbf{K}_{\mathbf{X}Z_c} \Psi_c^{-1} \mathbf{u}_i^c, \mathbf{K}_{\mathbf{XX}} - \sigma^{-2} \mathbf{K}_{\mathbf{X}Z_c} \Psi_c^{-1} \mathbf{K}_{Z_c\mathbf{X}}\right).$$

Collecting all the random variables in matrix forms, this leads to

$$p(U_c) = \mathcal{MN}(\mathbf{0}, \sigma_r^2 I, \Psi_c), \tag{29}$$
$$\begin{aligned} p(\mathbf{F} | \mathbf{X}, U_c) &= \mathcal{MN}\left(\sigma_c \sigma^{-2} U_c \Psi_c^{-1} \mathbf{K}_{Z_c\mathbf{X}}, \sigma_r^2 I, \sigma_r^{-2}(\mathbf{K}_{\mathbf{XX}} - \sigma^{-2} \mathbf{K}_{\mathbf{X}Z_c} \Psi_c^{-1} \mathbf{K}_{Z_c\mathbf{X}})\right) \\ &= \mathcal{MN}\left(U_c \Psi_c^{-1} \sigma_c Z_c \mathbf{X}, \sigma_r^2 I, \mathbf{X}^\top \sigma_c^2 (I - Z_c^\top \Psi_c^{-1} Z_c) \mathbf{X}\right). \end{aligned} \tag{30}$$

Also recall from conditioning rules of matrix normal distributions, we have that

$$p(W | U_c) = \mathcal{MN}\left(U_c \Psi_c^{-1} \sigma_c Z_c, \sigma_r^2 I, \sigma_c^2 (I - Z_c^\top \Psi_c^{-1} Z_c)\right).$$

Since for $W \sim \mathcal{MN}(\mathbf{M}, \Sigma_1, \Sigma_2)$ we have $W\mathbf{X} \overset{d}{\sim} \mathcal{MN}(\mathbf{MX}, \Sigma_1, \mathbf{X}^\top \Sigma_2 \mathbf{X})$, this immediately shows that $p(\mathbf{F} | \mathbf{X}, U_c)$ is the push-forward distribution of $p(W | U_c)$ for the operation $\mathbf{F} = W\mathbf{X}$. In other words:

$$\mathbf{F} \sim p(\mathbf{F} | \mathbf{X}, U_c) \quad \Leftrightarrow \quad W \sim p(W | U_c), \ \mathbf{F} = W\mathbf{X}.$$

This confirms the interpretation of U_c as "scaled noisy inducing outputs" that lies in the same space as $\mathbf{F}$. Notice that in the main text we provide a pictorial visualisation of U_c by selecting $\sigma_c = 1$. As the inducing weights U are the focus of our analysis here, we conclude that this specific choice of σ_c is without loss of generality.

So far the U_c variables assist the posterior inference to capture correlations across functions values of different inputs. Up to now the function values remain independent across output dimensions, which is also reflected by the diagonal row covariance matrices in the above matrix normal distributions. As in neural networks the output dimension can be fairly large (e.g. $d_{out} = 1000$), to further improve memory efficiency, we proceed to project the column vectors of U_c to an M_{out} dimensional space with $M_{out} << d_{out}$. This dimension reduction step is done with a generative approach, similar to probabilistic PCA (Tipping & Bishop, 1999):

$$U \sim \mathcal{MN}(0, \Psi_r, \Psi_c), \quad U_c | U \sim \mathcal{MN}(\sigma_r Z_r^\top \Psi_r^{-1} U, \sigma_r^2 (I - Z_r^\top \Psi_r^{-1} Z_r), \Psi_c). \tag{31}$$

Note that the column covariance matrices in the above two matrix normal distributions are the same, and the conditional sampling procedure is done by a linear transformation of the columns in U plus noise terms. Again from the marginalisation and conditioning rules of matrix normal distributions, we have that the full joint distribution (5), after proper marginalisation and conditioning, returns

$$p(U) = \mathcal{MN}(0, \Psi_r, \Psi_c), \quad p(U_c | U) = \mathcal{MN}(\sigma_r Z_r^\top \Psi_r^{-1} U, \sigma_r^2 (I - Z_r^\top \Psi_r^{-1} Z_r), \Psi_c).$$

This means U can be viewed as "projected noisy inducing points" for the GP $p(\mathbf{F})$, whose corresponding "inducing inputs" are row vectors in Z_c. Similarly, column vectors in $U_r\mathbf{X}$ can be viewed as the noisy projections of the column vectors in $\mathbf{F}$, in other words U_r can also be viewed as "neural network weights" connecting the data inputs $\mathbf{X}$ to the projected output space that U lives in.

As for the variational objective, since $q(W|U)$ and $p(W|U)$ only differ in the scale of the covariance matrices, it is straightforward to show that the push-forward distribution $q(W|U) \rightarrow q(\mathbf{F}|\mathbf{X}, U)$ has the same mean as $p(\mathbf{F}|\mathbf{X}, U)$, but with a different covariance matrix that scales $p(\mathbf{F}|\mathbf{X}, U)$'s covariance matrix by λ^2. As the operation $\mathbf{F} = W\mathbf{X}$ maps $W \in \mathbb{R}^{d_{out} \times d_{in}}$ to $\mathbf{F} \in \mathbb{R}^{d_{out} \times N}$, this means the conditional KL is scaled up/down, depending on whether $N \geq d_{in}$ or not:

$$\mathbb{KL}[q(\mathbf{F}|\mathbf{X}, U)||p(\mathbf{F}|\mathbf{X}, U)] = \frac{N}{d_{in}} R(\lambda), \quad R(\lambda) := \mathbb{KL}[q(W|U)||p(W|U)].$$

In summary, the push-forward distribution of $q(W_{1:L}, U_{1:L}) \rightarrow q(\mathbf{F}_{1:L}, U_{1:L})$ is

$$q(\mathbf{F}_{1:L}, U_{1:L}) = \textstyle\prod_{l=1}^{L} q(\mathbf{F}_l|\mathbf{F}_{l-1}, U_l)q(U_l), \quad \mathbf{F}_0 := \mathbf{X},$$

and the corresponding variational lower-bound for $q(\mathbf{F}_{1:L}, U_{1:L})$ becomes (with $\mathcal{D} = (\mathbf{X}, \mathbf{Y})$)

$$\mathcal{L}(q(\mathbf{F}_{1:L}, U_{1:L})) = \mathbb{E}_{q(\mathbf{F}_{1:L})}[\log p(\mathbf{Y}|\mathbf{F}_L)] - \textstyle\sum_{l=1}^{L} \left(\frac{N}{d_{in}^l} R(\lambda_l) + \mathbb{KL}[q(U_l)||p(U_l)] \right), \tag{32}$$

with d_{in}^l the input dimension of layer l.

Note that $\mathbb{E}_{q(\mathbf{F}_{1:L})}[\log p(\mathbf{Y}|\mathbf{F}_L)] = \mathbb{E}_{q(W_{1:L})}[\log p(\mathbf{Y}|\mathbf{X}, W_{1:L})] = \mathbb{E}_{q(W_{1:L})}[\log p(\mathcal{D}|W_{1:L})]$. Comparing equations (8) and (32), we see that the only difference between weight-space and function-space variational objectives comes in the scale of the conditional KL term. Though not investigated in the experiments, we conjecture that it could bring potential advantage to optimise the following variational lower-bound:

$$\tilde{\mathcal{L}}(q(\mathbf{F}_{1:L}, U_{1:L})) = \mathbb{E}_{q(\mathbf{F}_{1:L})}[\log p(\mathbf{Y}|\mathbf{F}_L)] - \textstyle\sum_{l=1}^{L} \left(\beta_l R(\lambda_l) + \mathbb{KL}[q(U_l)||p(U_l)] \right), \tag{33}$$
$$\beta_l = \min(1, \frac{N}{d_{in}^l}).$$

The intuition is that, as uncertainty is expected to be lower when $N \geq d_{in}$, it makes sense to use $\beta = 1 \leq N/d_{in}$ to reduce the regularisation effect introduced by the KL term. In other words, this allows the variational posterior to focus more on fitting the data, and in this "large-data" regime over-fitting is less likely to appear. On the other hand, function-space inference approaches (such as GPs) often return better uncertainty estimates when trained on small data ($N < d_{in}$). So choosing $\beta = N/d_{in} < 1$ in this case would switch to function-space inference and thereby improving uncertainty quality potentially. In the CIFAR experiments, the usage of convolutional filters leads to the fact that $N \geq d_{in}^l$ for all ResNet layers. Therefore in those experiments $\beta_l = 1$ for all layers, which effectively falls back to the weight-space objective (8).

E WHITENING AND HIERARCHICAL INDUCING VARIABLES

The inducing weights $U_{1:L}$ further allow for introducing a single inducing weight matrix U that is shared across the network. By doing so, correlations of weights between layers in the approximate posterior are introduced. The inducing weights are then sampled jointly conditioned on the global inducing weights. This requires that all inducing weight matrices are of the same size along at least one axis, such that they can be concatenated along the other one.

The easiest way of introducing a global inducing weight matrix is by proceeding similarly to the introduction of the per-layer inducing weights. As a pre-requisite, we need to work with "whitened" inducing weights, i.e. set the covariance of the marginal $p(U_l)$ to the identity and pre-multiply the covariance block between W_l and U_l with the inverse Cholesky of Ψ_l. In this whitened model, the full augmented prior per-layer is:

$$\begin{pmatrix} W & U_c \\ U_r & U \end{pmatrix} \sim p(W, U_c, U_r, U) := \mathcal{MN}(0, \tilde{\Sigma}_r, \tilde{\Sigma}_r), \tag{34}$$

$$\text{with} \quad \tilde{L}_r = \begin{pmatrix} \sigma_r I & 0 \\ L_r^{-1} Z_r & L_r^{-1} D_r \end{pmatrix} \quad \text{s.t.} \quad \tilde{\Sigma}_r = \tilde{L}_r \tilde{L}_r^\top = \begin{pmatrix} \sigma_r^2 I & \sigma_r Z_r^\top L_r^{-\top} \\ \sigma_r L_r^{-1} Z_r & I \end{pmatrix}$$

$$\text{and} \quad \tilde{L}_c = \begin{pmatrix} \sigma_c I & 0 \\ L_c^{-1} Z_c & L_c^{-1} D_c \end{pmatrix} \quad \text{s.t.} \quad \tilde{\Sigma}_c = \tilde{L}_c \tilde{L}_c^\top = \begin{pmatrix} \sigma_c^2 I & \sigma_c Z_c^\top L_c^{-\top} \\ \sigma_c L_c^{-1} Z_c & I \end{pmatrix}.$$

One can verify that this whitened model leads to the same conditional distribution $p(W|U)$ as presented in the main text. After whitening, for each U_l we have that $p(\text{vec}(U_l)) = \mathcal{N}(0, I)$, therefore we can also write their joint distribution as $p(\text{vec}(U_{1:L})) = \mathcal{N}(0, I)$. In order to construct a matrix normal prior $p(U_{1:L}) = \mathcal{MN}(0, I, I)$, the inducing weight matrices $U_{1:L}$ needs to be stacked either along the rows or columns, requiring the other dimension to be matching across all layers. Then, As the covariance is the identity with $\sigma = \sigma_r = \sigma_c = 1$, we can augment $p(U_{1:L})$ in the exact same way as we previously augmented the prior $p(W_l)$ with U_l.

F Experimental details

F.1 Regression experiments

Following (Foong et al., 2019), we sample 50 inputs each from $\mathcal{U}\,[-1, -0.7]$ and $\mathcal{U}\,[0.5, 1]$ as inputs and targets $y \sim \mathcal{N}(\cos(0.4x + 0.8), 0.01)$. As a prior we use a zero-mean Gaussian with standard deviation $\frac{4}{\sqrt{d_{in}}}$ for the weights and biases of each layer. Our network architecture has a single hidden layer of 50 units and uses a $\tanh$-nonlinearity. All three variational methods are optimised using Adam (Kingma & Ba, 2014) for $20,000$ updates with an initial learning rate of 10^{-3}. We average over 32 MC samples from the approximate posterior for every update. For Ensemble-U and FCG-U we decay the learning rate by a factor of 0.1 after $10,000$ updates and the size of the inducing weight matrix is 2×25 for the input layer (accounting for the bias) and 25×1 for the output layer. Ensemble-U uses an ensemble size of 8.

For NUTS we use the implementation provided in Pyro (Bingham et al., 2019). We draw a total of $25,000$ samples, discarding the first 5000 as burn-in and using 1000 randomly selected ones for prediction.

F.2 Classification experiments

We base our implementation on the Resnet-18 class in torchvision (Paszke et al., 2017), replacing the input convolutional layer with a 3×3 kernel size and removing the max-pooling layer. We train the deterministic network on CIFAR-10 using Adam with a learning rate of 3×10^{-4} for 200 epochs. On CIFAR-100 we found SGD with a momentum of 0.9 and initial learning rate of 0.1 decayed by a factor of 0.1 after 60, 120 and 160 epochs to lead to better accuracies. The ensemble is formed of the five deterministic networks trained with different random seeds.

For FFG-W we initialised the mean parameters using the default initialisation in pytorch for the corresponding deterministic layers. The initial standard deviations are set to 10^{-4}. We train using Adam for 200 epochs on CIFAR-10 with a learning rate of 3×10^{-4}, and 300 epochs on CIFAR-100 with an initial learning rate of 10^{-3}, decaying by a factor of 0.1 after 200 epochs. On both datasets we only use the negative log likelihood part of the variational lower bound for the first 100 epochs as initialisation to the maximum likelihood parameter and then anneal the weight of the kl term linearly over the following 50 epochs. For the prior we use a standard Gaussian on all weights and biases and restrict the standard deviation of the posterior to be at most $\sigma_{max} = 0.1$. We also experimented with a larger upper limit of $\sigma_{max} = 0.3$, but found this to negatively affect both accuracy and calibration.

All the U-space approaches use Gaussian priors $p(\text{vec}(W_l)) = \mathcal{N}(0, 1/\sqrt{d_{in}})$, motivated by the connection to GPs. Hyperparameter and optimisation details for the inducing weight methods on CIFAR-10 are discussed below in the details on the ablation study. On CIFAR-100, we use the same optimisation settings as for FFG-W, but train for an extra 50 epochs for Ensemble-U as the optimisation had not converged after 300 epochs. For the tables and figures in the main text, we set

$\lambda_{max} = 0.1$ for Ensemble-U on both dataset, and $\sigma_{max} = 0.1, \lambda_{max} = 0.03$ on CIFAR-10 and $\sigma_{max} = 0.03, \lambda_{max} = 0.1$ on CIFAR-100 for FFG-U. We initialise the entries of the Z matrices by sampling from a zero-mean Gaussian with variance $\frac{1}{M}$ and set the diagonal entries of the D matrices to 10^{-3}. For FFG-U we initialise the mean of the variational Gaussian posterior in U-space by sampling from a standard Gaussian and set the initial variances to 10^{-3}. For Ensemble-U initialisation, we draw an $M \times M$ shaped sample from a standard Gaussian that is shared across ensemble members and add independent Gaussian noise with a standard deviation of 0.1 for each member. We use an ensemble size of 5. During optimisation, we draw 5 MC samples per update step for both FFG-U and Ensemble-U (such that each ensemble member is used once). For testing we use 20 MC samples for all variational methods. We fit BatchNorm parameters by minimising the negative log likelihood.

The study of hyper-parameter selection on CIFAR-10 We run the inducing weight method with the following options:

- Row/column dimensions of U_l (M): $M \in \{16, 32, 64, 128\}$.
  We set $M = M_{in} = M_{out}$ except for the last layer, where we use $M_{in} = M$ and $M_{out} = 10$.
- λ_{max} values for FFG-U and Ensemble-U: $\lambda_{max} \in \{0, 0.03, 0.1, 0.3\}$.
  When $\lambda_{max} = 0$ it means $q(W|U)$ is a delta measure centered at the mean of $p(W|U)$.
- $\sigma_{\max}$ values for FFG-U: $\sigma_{\max} = \{0, 0.1, 0.3\}$.
  When $\sigma_{max} = 0$ we use a MAP estimate for U.

Each experiment is repeated with 5 random seeds to collect the averaged results. The models are trained with 100 epochs in total. We first run 50 epochs of maximum likelihood to initialise the model, then run 40 epochs training on the modified variational lower-bound with KL annealing (linear scaling schedule), finally we run 10 epochs of training with the variational lower-bound (i.e. no KL annealing). We use Adam optimiser with learning rate $3e-4$ and the default β_1, β_2 parameters in pytorch's implementation.

G Additional Results

Below we provided extended versions of Table 2 for CIFAR-10 (Table 4) and CIFAR-100 (Table 5). These tables contain standard errors across the random seeds for the corresponding metrics and we additionally report negative log likelihoods (NLL) and Brier scores. The error bar results are not available for Ensemble-W, as it is constructed by 5 independently trained deterministic neural network with maximum likelihood.

Table 4: CIFAR-10 complete in-distribution results

	CIFAR-10			
Method	Acc. (%)↑	NLL ↓	ECE (%)↓	Brier ↓
Deterministic	93.02±0.10	0.42±0.01	5.23±0.10	0.12±0.00
Ensemble-W	94.94	0.18	1.25	0.08
FFG-W	93.22±0.03	0.20±0.00	0.55±0.04	0.10±0.00
FFG-U	91.52±0.12	0.26±0.00	1.31±0.12	0.12±0.00
Ensemble-U	92.20±0.11	0.24±0.00	0.80±0.04	0.11±0.00

The number of parameters for FFG-U in the ResNet-18 experiments is reported in Table 6.

In Tables 7-10 we report the numerical results for Figure 6.

Table 5: CIFAR-100 complete in-distribution results

	CIFAR-100			
Method	Acc. (%)↑	NLL ↓	ECE (%)↓	Brier ↓
Deterministic	72.68±0.20	1.80±0.01	19.41±0.19	0.45±0.00
Ensemble-W	76.61	1.10	6.25	0.34
FFG-W	73.44±0.13	1.10±0.01	5.49±0.06	0.37±0.00
FFG-U	75.69±0.26	0.96±0.02	5.20±0.46	0.34±0.00
Ensemble-U	76.10±0.18	0.94±0.01	2.49±0.12	0.33±0.00

Table 6: Parameter counts for the FFG-U models with different U size M.

M	$M = 16$	$M = 32$	$M = 64$	$M = 128$	$M = 256$	Deterministic
Abs. value	602261	1216045	2509205	5352853	12072341	11173962
relative size (%)	5.39	10.88	22.46	47.90	108.04	100

Table 7: Corrupted CIFAR-10 accuracy (↑) values (in %).

	Skew Intensity				
Method	1	2	3	4	5
Deterministic	85.12±2.46	78.35±3.08	71.95±4.09	64.47±5.15	54.51±5.30
Ensemble-W	88.89±2.24	83.33±2.67	77.73±3.61	70.56±4.62	60.05±4.98
FFG-W	84.63±2.14	77.86±2.45	71.01±3.31	62.76±4.17	52.02±4.51
FFG-U	83.42±2.48	76.72±3.02	71.18±3.72	64.14±4.44	54.04±4.44
Ensemble-U	85.19±2.21	79.06±2.77	73.28±3.71	67.16±4.39	57.34±4.38

Table 8: Corrupted CIFAR-10 ECE (↓) values (in %).

	Skew Intensity				
Method	1	2	3	4	5
Deterministic	11.70±2.11	17.39±2.70	22.99±3.71	29.74±4.71	38.27±4.98
Ensemble-W	2.81±1.03	4.71±1.28	7.37±2.13	11.23±2.84	16.89±3.23
FFG-W	13.37±0.64	12.36±0.83	10.89±0.89	10.36±1.00	11.61±1.43
FFG-U	4.56±1.39	7.87±1.78	10.99±2.40	15.33±2.86	21.65±3.03
Ensemble-U	2.88±0.96	5.65±1.53	9.04±2.45	12.25±2.92	17.31±3.06

Table 9: Corrupted CIFAR-100 accuracy (↑) values (in %).

	Skew Intensity				
Method	1	2	3	4	5
Deterministic	63.12±2.24	55.73±2.61	50.58±3.19	44.05±3.61	34.50±3.38
Ensemble-W	67.32±2.37	59.77±2.78	54.46±3.45	47.70±4.00	37.78±3.75
FFG-W	58.46±2.92	48.76±3.57	43.12±4.19	36.77±4.27	27.93±3.72
FFG-U	63.56±3.11	54.83±3.76	49.44±4.39	43.27±4.64	33.41±4.06
Ensemble-U	62.55±3.39	53.48±4.22	48.30±4.68	42.38±4.74	32.75±3.95

Table 10: Corrupted CIFAR-100 ECE (↓) values (in %).

	Skew Intensity				
Method	1	2	3	4	5
Deterministic	26.36±1.75	31.83±2.00	35.88±2.49	41.14±2.94	48.91±2.84
Ensemble-W	9.79±1.27	13.01±1.43	15.85±1.89	19.69±2.39	25.42±2.35
FFG-W	12.69±0.62	10.69±0.77	11.09±0.95	11.21±1.31	11.76±1.74
FFG-U	8.99±1.75	13.29±2.24	16.65±2.97	20.46±3.46	25.76±3.17
Ensemble-U	7.56±1.65	12.08±2.39	14.82±2.79	17.60±2.88	22.61±2.46