Try Depth Instead of Weight Correlations: Mean-field is a Less Restrictive Assumption for Variational Inference in Deep Networks

Sebastian Farquhar
OATML, Department of Computer Science
University of Oxford
sebastian.farquhar@cs.ox.ac.uk

Lewis Smith
OATML, Department of Computer Science
University of Oxford
lsgs@robots.ox.ac.uk

Yarin Gal
OATML, Department of Computer Science
University of Oxford
yarin.gal@cs.ox.ac.uk

1 Introduction

Ever since variational inference was introduced for Bayesian neural networks, researchers have assumed that the 'mean-field' approximation—that the posterior over the weights has diagonal covariance—was a major limitation [Barber and Bishop, 1998]. This assumption continues to drive research into tractable non-diagonal approximations for the covariance of the approximating posterior to this day (e.g., [Louizos and Welling, 2016, Sun et al., 2017, Oh et al., 2019]).

Here we show that while the mean-field approximation is restrictive in shallow networks [Foong et al., 2019], it is less restrictive in deep networks. More concretely, we show that reparameterizing a single linear transformation W as the product of **three or more** linear transformations $W = ABC$, where each of the transformations A, B, C is approximated with a mean-field distribution, induces a non-diagonal covariance over W with all entries potentially non-zero. Increasing depth further allows for more flexible covariance matrices by increasing the number of parameters which construct the covariance matrix. We derive this covariance matrix, and argue that the principles extend when non-linearities are introduced between the linear transformations. We evaluate this claim empirically in neural networks with ReLU non-linearities, where we demonstrate that the divergence between a diagonal- and full-covariance approximation to the posterior falls as model depth grows.

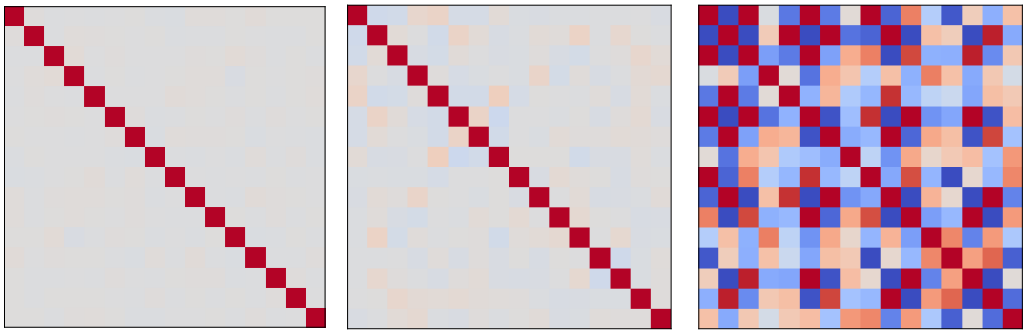
Our work challenges the long-held assumption that progress in variational inference requires computationally tractable ways of enabling correlation between weights in the same layer. With this insight, researchers can focus on improving the training of deep models with mean-field variational inference, rather than building computationally expensive non-mean-field approximations. This also highlights the importance of replacing the standard UCI experimental settings for comparison of Bayesian deep learning methods. UCI experiments focus on models with one hidden layer, obscuring properties of deeper models which might differ greatly.

2 Developing Intuition Using Deep Linear Models

Although we are most interested in neural networks that include non-linearities, we begin by analysing linear neural networks, before considering non-linearities in sections 3 and 4.

In this section, we develop intuition using linear networks in two ways. First, we work through a simplified toy example and show that a deep linear network composed of 'mean-field' layers can be equivalent to a single layer with a non-diagonal covariance matrix. Second, we derive the covariance matrix of a deep linear neural network composed of individual layers with diagonal covariance

4th workshop on Bayesian Deep Learning (NeurIPS 2019), Vancouver, Canada.



(a) 1-layer mean-field. (b) Product of 5 mean-field layers. (c) Product of 20 mean-field layers.

Figure 1: Covariance matrix for simulated deep linear product matrix composed of randomly chosen mean-field layers. Redder is more positive, bluer more negative. (a) For 1 layer, covariance is mean-field. (b,c) For deeper models, the mean-field layers induce a full-covariance product matrix.

matrices and visualize this with numerical simulations. We prove that **only three layers** create a covariance matrix which captures correlations between every pair of weights in the product W.

2.1 Intuition

Consider a linear model with a 2x2 weight matrix, two input features and two outputs. We can choose to reparameterize the weight matrix as the product of two matrices, A and B, and write down the values of the elements of the product matrix in terms of the elements of A and B:

$$\mathbf{o} = BA\mathbf{x} = \mathbf{x} = W^{(2)}\mathbf{x} = \begin{pmatrix} B_{11}A_{11} + B_{12}A_{21} & B_{11}A_{12} + B_{12}A_{22} \\ B_{21}A_{11} + B_{22}A_{21} & B_{21}A_{12} + B_{22}A_{22} \end{pmatrix} \mathbf{x} \tag{1}$$

Suppose we make the mean-field approximation for each layer, so each weight in A and B is correlated only with itself. The resulting covariance matrix of $W^{(2)}$ is not diagonal because elements share the same random variables. For example, B_{11} is in each entry in the top row, so they have non-zero covariance. That is to show: we can create a product matrix with non-diagonal covariance by stacking mean-field layers, in a linear setting.

2.2 Numerical Simulation of Linear Product Matrices

In order to develop our intuition further, we visualize the covariance matrices for the product of multiple mean-field 4x4 layers. Every layer in the model makes the mean-field assumption: the covariance matrix is assumed diagonal and weights are sampled independently from a Gaussian distribution with a mean uniformly chosen between -0.1 and 0.1 and a standard deviation of 0.5. We sample 10,000 realizations of each layer, and calculate the resulting product matrix, 'flattening' each network into a single layer like in equation (1), which is possible only in the linear case. In Figure 1, we show the simulated covariance matrix for product matrices of different depths as a heatmap (redder cells show stronger positive correlation, while bluer cells show stronger anti-correlation). For the 'product' matrix from a single layer, the covariance matrix is obviously mean-field (Figure 1 (a)). For deeper networks, there are non-zero correlations for off-diagonal elements.

Note, of course, that the resulting distributions over the weights of the product matrix are not necessarily Gaussian.

2.3 Deriving the Covariance Matrix for Fully Connected Linear Networks

Now we develop an expression for the covariance matrix of the product of multiple linear layers. We show that three layers are sufficient for non-zero covariances between all weights in the product matrix, so long as the layer means are non-zero.

Consider the same setting as in section 2.1, except that the two hidden layers may be arbitrarily large, with K units. We can 'flatten' multiple layers into a single product matrix like in equation (1) and

express this in index notation:

$$W^{(2)}_{ab} = \sum_{i=1}^{K} A_{ai} B_{ib}. \tag{2}$$

We can further write the mean-field assumption in index notation:

$$\mathrm{Cov}(A_{ij}, A_{kl}) = \delta_{ik}\delta_{jl}\sigma^A_{ij}, \quad \mathrm{Cov}(B_{ij}, B_{kl}) = \delta_{ik}\delta_{jl}\sigma^B_{ij}. \tag{3}$$

We want the general expression for the covariance matrix of the product of the two matrices: $\mathrm{Cov}(W^{(2)}_{ab}, W^{(2)}_{cd})$. Through linearity we reduce the covariance to the sum of covariances of scalar terms, and rewrite using known covariances. Full derivations for this section are provided in the appendix A, which show that the covariance in the case of the product of two mean-field matrices is:

$$\mathrm{Cov}(W^{(2)}_{ab}, W^{(2)}_{cd}) = \sum_{ij} \delta_{ij} \Big(\delta_{ac}\delta_{bd}\sigma^A_{ai}\sigma^B_{ib} + \mu^A_{ai}\mu^A_{cj}\delta_{bd}\sigma^B_{ib} + \mu^B_{ib}\mu^B_{jd}\delta_{ac}\sigma^A_{ai} \Big).$$

The first term represents non-zero covariance for parameters with themselves, the second term is non-zero covariance when parameters share a column, and the third when they share a row. Note that the matrix variate Gaussian (MVG) distribution used by Louizos and Welling [2016] and Sun et al. [2017] is similar in that MVG also supports correlations along the rows and columns of the weight covariance matrix (though the parameterization is different).

Note that our derivation does not assume that any distributions are Gaussian, although these are commonly used in Bayesian deep learning.

This is not yet a 'full' covariance (there exist pairs of weights with zero correlation). But let us consider the product of three mean-field linear layers, $W^{(3)} = CW^{(2)}$ and the resulting covariance $\mathrm{Cov}(W^{(3)}_{ab}, W^{(3)}_{cd}) = \mathrm{Cov}(W^{(2)}_{ai}C_{ib}, W^{(2)}_{cj}C_{jd})$. By iterating our result from equation (4), we find:

$$\begin{aligned}\mathrm{Cov}(W^{(3)}_{ab}, W^{(3)}_{cd}) = \sum_{ij} \delta_{ij} \Bigg(& \sum_{kl} \delta_{kl} \Big(\delta_{ac}\delta_{ij}\sigma^A_{ak}\sigma^B_{ki} + \mu^A_{ak}\mu^A_{cl}\delta_{ij}\sigma^B_{ki} + \mu^B_{ki}\mu^B_{lj}\delta_{ac}\sigma^A_{ak} \Big)\delta_{bd}\sigma^C_{ib} \\ & + \mathbb{E}[W^{(2)}_{ai}]\,\mathbb{E}[W^{(2)}_{cj}]\delta_{bd}\sigma^C_{ib} \\ & + \mu^C_{ib}\mu^C_{jd} \sum_{kl} \delta_{kl} \Big(\delta_{ac}\delta_{ij}\sigma^A_{ak}\sigma^B_{ki} + \mu^A_{ak}\mu^A_{cl}\delta_{ij}\sigma^B_{ki} + \mu^B_{ki}\mu^B_{lj}\delta_{ac}\sigma^A_{ak} \Big) \Bigg).\end{aligned} \tag{4}$$

After three layers, therefore, we have a matrix $W^{(3)}$ which has non-zero correlations between all weight pairs, because the red terms have no Kronecker delta functions in the indices over the outside of the $W^{(3)}$ matrix (a, b, c, and d). This means that, in the linear case at least, rather than using a complicated posterior that explicitly models weight covariance and may require matrix inversion or inducing point approximations, we can just use a deeper linear network!

Note that for the product matrix $W^{(N)}$, composed of N layers, we have only $(2N-1)K$ free parameters[1], which are jointly parameterizing K^2 covariance elements with $\frac{K(K+1)}{2}$ degrees of freedom. The product matrix therefore does impose structure on the approximate posterior's covariance, although the approximation becomes richer as the number of layers increases.

3 Extending to Deep Neural Networks

What does this mean for neural networks which have non-linearities between their linear transformations? Here we cannot simply derive the covariance of product matrix elements, because there is no product matrix. However, note that the non-zero correlations between W's weight pairs (the induced product matrix) are related to the correlations between the *feature vectors* in the deep linear model. This intuition extends to the non-linear case as well, where we wish to study how the units correlate with each other. The reason that the mean-field covariance in three or more layers gives

[1]The means also contribute to the covariance terms, but one set of them are not free variables.

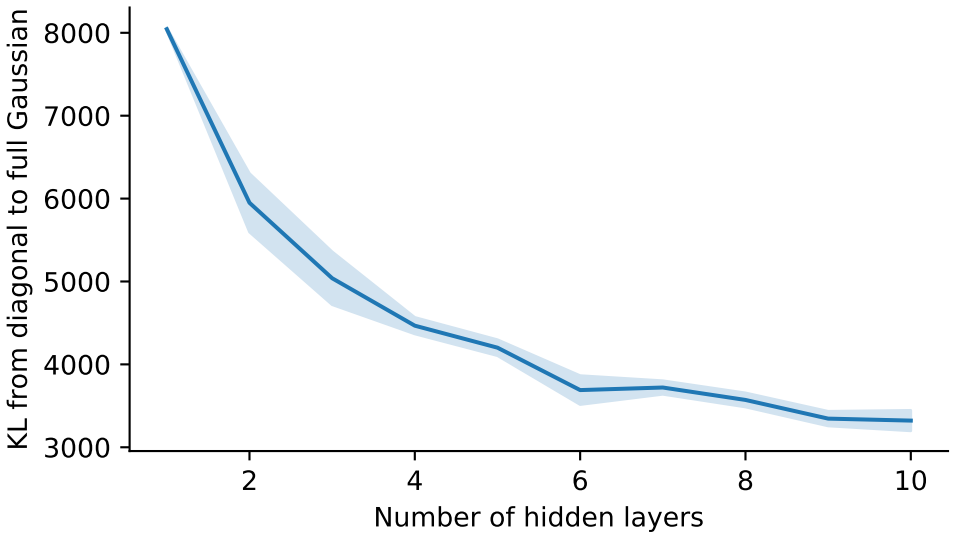


Figure 2: Deeper models pay a smaller price for the 'mean-field' assumption. Figure shows the KL divergence between q_{diag}—a diagonal-covariance Gaussian approximation to the true posterior—and q_{full}—a full-covariance Gaussian approximation. A smaller $KL\big(q_{\text{diag}} \parallel q_{\text{full}}\big)$ reflects a diagonal distribution that is closer to the full-covariance distribution. All models have roughly the same number of parameters (1000) in order to ensure that observations do not just reflect the effect of overparameterized models. Shading reflects the standard error of the mean from 20 runs.

rise to such complex combined behaviour is that after the first two layers the inputs to the next layer are all *already correlated*—given the structure of a neural network, all units depend on *all* of the weights preceding the layer before them. The non-linearity is an element-wise operation which does not change which units depend on which previous weights.

We hypothesise that with sufficient depth, there is at least one mode of the true posterior over the neural network weights which is 'nearly' diagonal. This is because the network can model an induced covariance between outputs conditioned on inputs through shared dependence on 'upstream' random variables, rather than correlated random variables This observation is inversely related to Dropout as a Bayesian approximation and Variational Dropout [Gal, 2016, Kingma et al., 2015] where a variational distribution was placed over the activations (units / feature vectors), and the posterior over the weights was *implicit*. We test this hypothesis empirically in the next section, by identifying a mode of the true posterior which is discoverable by mean-field variational inference and fitting both diagonal- and full-covariance Gaussian approximations to it.

4 Experiments

We validate our theoretical observations in the linear case using experiments in neural networks with ReLU non-linearities. In this experiment, our goal is to compare a full-covariance Gaussian approximation of the true posterior to an approximate posterior that imposes a mean-field constraint. We find that deeper networks pay less of a 'price' for the mean-field assumption (see Figure 2). We show that the distance between a full-covariance Gaussian approximate posterior and its closest diagonal approximation falls as the network depth increases. This suggest that, even with non-linearities, depth makes the mean-field assumption less restrictive. We hypothesize that this is because the true posterior a deeper network will have many modes around different possible arrangements of weights, and that in a deeper network at least some of these can have approximately diagonal covariance matrices.

We consider neural networks of different depths, with their widths set such that they all have approximately 1,000 parameters. The models are trained to classify points from the 'two-moons'

data distribution with noise 0.1 and 10,000 datapoints, and all depths achieve nearly perfect accuracy on test data.

For each network, we first pre-train using MFVI and use these weights to initialize Hamiltonian Monte Carlo (HMC) [Neal, 1995] near a mode that MFVI would discover. We then draw samples from the true posterior using HMC with the No U-turn Sampler [Hoffman and Gelman, 2014] (full implementation details in Appendix B). To estimate the full covariance Gaussian, we calculate the empirical covariance matrix using 1000 samples from the true posterior. Call this full-covariance Gaussian approximation to the true posterior $q_{\text{full}} \coloneqq \mathcal{N}(\mu, \Sigma)$. We then find a diagonal approximation to the full-covariance approximate posterior, $q_{\text{diag}} \coloneqq \mathcal{N}(\mu', \Sigma')$. We pick μ' and Σ' to minimize $\mathrm{KL}\big(q_{\text{diag}} \parallel q_{\text{full}}\big)$. The analytical solution for μ' and Σ' is:

$$\mu' = \mu \tag{5}$$

$$\Sigma' = \mathrm{diag}(\Sigma^{-1})^{-1}. \tag{6}$$

The $\mathrm{KL}\big(q_{\text{diag}} \parallel q_{\text{full}}\big)$ then represents the closest possible distance between a mean-field Gaussian and a full-covariance Gaussian approximation of the true posterior. The fact that the KL falls as the model gets deeper suggests that the diagonal covariance Gaussian becomes an increasingly good approximation to the full-covariance Gaussian in deeper models.

Note that the full-covariance Gaussian we learn computes the covariance between all parameters in the model, not only within a single layer, as is commonly considered. This supports our hypothesis that the model needs less covariance between weights because it can model covariance between outputs using shared dependence on earlier units, rather than covariance between weights.

5 Implications

Our work does not imply that sophisticated posterior approximations with structured covariances can never be useful, but it does suggest that they are not a *necessary* requirement for good predictive performance in BNNs. Researchers have taken as assumed knowledge that the mean-field approximation is a severe restriction, true in shallow networks, but not necessarily in deep ones. The experimental setting of the popular UCI benchmark for Bayesian deep learning, using a single hidden layer, seems to exacerbate this problem. Focusing on UCI, researchers try to make weight correlations in single layers as expressive as possible, leading to models which are too expensive to train on anything bigger than UCI, perpetuating the cycle. Given the insights from this paper, we hope that future research can work on identifying and fixing other problems with deep mean-field variational inference, rather than trying to craft expensive expressive correlations in shallow networks.

6 Acknowledgements

We would like to thank Adam Cobb for his help applying his Hamiltorch package [Cobb et al., 2019] and tuning hyperparameters for HMC. We would also like to thank Wendelin Boehmer, Gregory Farquhar, Raza Habib, Clare Lyle, and Hippolyt Ritter for their comments and conversations.

This work was supported by the EPSRC CDT for Autonomous Intelligent Machines and Systems and for Cyber Security at the University of Oxford. It was also supported by the Alan Turing Institute.

References

David Barber and Christopher M Bishop. Ensemble Learning in Bayesian Neural Networks. page 20, 1998. 1

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. *Proceedings of the 32nd International Conference on Machine Learning*, 37: 1613–1622, 2015. B

Adam D. Cobb, Atılım Güneş Baydin, Andrew Markham, and Stephen J. Roberts. Introducing an Explicit Symplectic Integration Scheme for Riemannian Manifold Hamiltonian Monte Carlo. *arXiv:1910.06243 [cs, stat]*, October 2019. arXiv: 1910.06243. 6, B

Andrew Y. K. Foong, David R. Burt, Yingzhen Li, and Richard E. Turner. Pathologies of Factorised Gaussian and MC Dropout Posteriors in Bayesian Neural Networks. *arXiv:1909.00719 [cs, stat]*, September 2019. URL `http://arxiv.org/abs/1909.00719`. arXiv: 1909.00719. 1

Yarin Gal. Uncertainty in Deep Learning. *PhD Thesis*, 2016. 3

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CVPR*, 7(3):171–180, 2016. B

Matthew D Hoffman and Andrew Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15:1593–1623, 2014. 4

Diederik P. Kingma, Tim Salimans, and Max Welling. Variational Dropout and the Local Reparameterization Trick. *Advances In Neural Information Processing Systems*, pages 2575–2583, 2015. 3

Christos Louizos and Max Welling. Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors. *International Conference on Machine Learning*, pages 1708–1716, 2016. 1, 2.3

Radford M. Neal. *Bayesian learning for neural networks*. PhD Thesis, University of Toronto, 1995. 4

Changyong Oh, Kamil Adamczewski, and Mijung Park. Radial and Directional Posteriors for Bayesian Neural Networks. *arXiv*, February 2019. arXiv: 1902.02603. 1

Shengyang Sun, Changyou Chen, and Lawrence Carin. Learning Structured Weight Uncertainty in Bayesian Neural Networks. *Artificial Intelligence and Statistics*, pages 1283–1292, 2017. 1, 2.3

A Full Derivation of the Covariance Matrix for a Fully Connected Linear Network

Consider the setting of the toy illustration, except that the two hidden layers may be arbitrarily large, with size K. For simplicity, we consider index notation, so rewriting equations (1) and the definition of the mean-field approximation for the covariance:

$$W^{(2)}_{ab} = \sum_{i=1}^{K} A_{ai} B_{ib} \quad \text{and} \quad \mathrm{Cov}(A_{ij}, A_{kl}) = \delta_{ik}\delta_{jl}\sigma^A_{ij}, \quad \mathrm{Cov}(B_{ij}, B_{kl}) = \delta_{ik}\delta_{jl}\sigma^B_{ij}. \tag{7}$$

We then want to find the general expression for the covariance matrix of the product of the two matrices: $\mathrm{Cov}(W^{(2)}_{ab}, W^{(2)}_{cd})$. We can simplify this significantly using the fact that the covariance of sums of independent random variables is the sum of the covariances.

$$\begin{aligned}
\mathrm{Cov}(W^{(2)}_{ab}, W^{(2)}_{cd}) &= \mathrm{Cov}(\sum_i A_{ai} B_{ib}, \sum_j A_{cj} B_{jd}) && (8)\\
&= \sum_i \mathrm{Cov}(A_{ai} B_{ib}, \sum_j A_{cj} D_{jd}) && (9)\\
&= \sum_{ij} \mathrm{Cov}(A_{ai} B_{ib}, A_{cj} B_{jd}). && (10)
\end{aligned}$$

From the definition of covariance, we can rewrite this as:

$$\mathrm{Cov}(W^{(2)}_{ab}, W^{(2)}_{cd}) = \sum_{ij} \mathbb{E}[A_{ai} B_{ib} A_{cj} B_{jd}] - \mathbb{E}[A_{ai} B_{ib}]\,\mathbb{E}[A_{cj} B_{jd}] \tag{11}$$

And this can in turn be simplified using the fact that the A and B layers are independent of each other.

$$\mathrm{Cov}(W^{(2)}_{ab}, W^{(2)}_{cd}) = \sum_{ij} \mathbb{E}[A_{ai} A_{cj}]\,\mathbb{E}[B_{ib} B_{jd}] - \mathbb{E}[A_{ai}]\,\mathbb{E}[A_{cj}]\,\mathbb{E}[B_{ib}]\,\mathbb{E}[B_{jd}]. \tag{12}$$

We can now rewrite this in order to expose the structure of the covariance:

$$\begin{aligned}
\mathrm{Cov}(W^{(2)}_{ab}, W^{(2)}_{cd}) = \sum_{ij} &\Big(\mathbb{E}[A_{ai} A_{cj}] - \mathbb{E}[A_{ai}]\,\mathbb{E}[A_{cj}]\Big) \cdot \Big(\mathbb{E}[B_{ib} B_{jd}] - \mathbb{E}[B_{ib}]\,\mathbb{E}[B_{jd}]\Big)\\
&+ \mathbb{E}[A_{ai}]\,\mathbb{E}[A_{cj}]\Big(\mathbb{E}[B_{ib} B_{jd}] - \mathbb{E}[B_{ib}]\,\mathbb{E}[B_{jd}]\Big)\\
&+ \mathbb{E}[B_{ib}]\,\mathbb{E}[B_{jd}]\Big(\mathbb{E}[A_{ai} A_{cj}] - \mathbb{E}[A_{ai}]\,\mathbb{E}[A_{ck}]\Big).
\end{aligned} \tag{13}$$

Which can be made explicit as:

$$\begin{aligned}
\mathrm{Cov}(W^{(2)}_{ab}, W^{(2)}_{cd}) = \sum_{ij} &\mathrm{Cov}(A_{ai}, A_{cj})\mathrm{Cov}(B_{ib}, B_{jd})\\
&+ \mathbb{E}[A_{ai}]\,\mathbb{E}[A_{cj}]\mathrm{Cov}(B_{ib}, B_{jd})\\
&+ \mathbb{E}[B_{ib}]\,\mathbb{E}[B_{jd}]\mathrm{Cov}(A_{ai}, A_{cj}).
\end{aligned} \tag{14}$$

We can reduce this using our earlier definitions of these covariances based on the mean-field approximation in equation (7).

$$\mathrm{Cov}(W^{(2)}_{ab}, W^{(2)}_{cd}) = \sum_{ij} \delta_{ij}\Big(\delta_{ac}\delta_{bd}\sigma^A_{ai}\sigma^B_{ib} + \mu^A_{ai}\mu^A_{cj}\delta_{bd}\sigma^B_{ib} + \mu^B_{ib}\mu^B_{jd}\delta_{ac}\sigma^A_{ai}\Big).$$

The first term means non-zero covariance for parameters with themselves, the second term is non-zero covariance when parameters share a column, and the third when they share a row (with magnitudes depending on the mean parameter of each weight). Our covariance matrix now has $3K - 2$ non-zero entries rather than the K elements from a single layer of mean-field approximation.

But now let us consider the product of three mean-field linear layers, $W^{(3)} = CW^{(2)}$. We are now interested in the covariance $\mathrm{Cov}(W^{(3)}_{ab}, W^{(3)}_{cd}) = \mathrm{Cov}(W^{(2)}_{ai} C_{ib}, W^{(2)}_{cj} C_{jd})$. From equation (15, we have:

$$\begin{aligned}\mathrm{Cov}(W^{(2)}_{ai} C_{ib}, W^{(2)}_{cj} C_{jd}) = \sum_{ij} &\mathrm{Cov}(W^{(2)}_{ai}, W^{(2)}_{cj})\mathrm{Cov}(C_{ib}, C_{jd}) \\ &+ \mathbb{E}[W^{(2)}_{ai}]\,\mathbb{E}[W^{(2)}_{cj}]\mathrm{Cov}(C_{ib}, C_{jd}) \\ &+ \mathbb{E}[C_{ib}]\,\mathbb{E}[C_{jd}]\mathrm{Cov}(W^{(2)}_{ai}, W^{(2)}_{cj}). \end{aligned} \tag{15}$$

And now we can plug in our result from equation (15), remembering that the sums are over different indices, to get the resulting covariance matrix for $W^{(3)}$:

$$\begin{aligned}\mathrm{Cov}(W^{(3)}_{ab}, W^{(3)}_{cd}) = \sum_{ij} \delta_{ij} \Bigg(\sum_{kl} \delta_{kl} \Big(&\delta_{ac}\delta_{ij}\sigma^A_{ak}\sigma^B_{ki} + \mu^A_{ak}\mu^A_{cl}\delta_{ij}\sigma^B_{ki} + \mu^B_{ki}\mu^B_{lj}\delta_{ac}\sigma^A_{ak} \Big)\delta_{bd}\sigma^C_{ib} \\ &+ \mathbb{E}[W^{(2)}_{ai}]\,\mathbb{E}[W^{(2)}_{cj}]\delta_{bd}\sigma^C_{ib} \qquad (16) \\ &+ \mu^C_{ib}\mu^C_{jd} \sum_{kl} \delta_{kl} \Big(\delta_{ac}\delta_{ij}\sigma^A_{ak}\sigma^B_{ki} + \mu^A_{ak}\mu^A_{cl}\delta_{ij}\sigma^B_{ki} + \mu^B_{ki}\mu^B_{lj}\delta_{ac}\sigma^A_{ak} \Big) \Bigg). \qquad (17)\end{aligned}$$

After three layers, therefore, we have a matrix $W^{(3)}$ which has non-diagonal covariance matrix with K^2 non-zero elements, because the red terms have no Kronecker delta functions in the indices over the outside of the $W^{(3)}$ matrix (a, b, c, and d).

B Full Experimental Settings

For each model depth we use the largest width that keeps the number of weights in the model under 1000. We do this in order to have the fairest possible comparison between models—otherwise the properties we evaluate might only be because the deeper models have many more parameters, not because of the depth.

To pretrain the weights, we perform mean-field variational inference using the local reparameterization trick in the manner of Blundell et al. [2015], but we do not reweight the KL-divergence and we use unit Gaussian priors over the weights. We initialize using the He Kaiming scheme [He et al., 2016] and set the initial variance parameters, ρ, which are transformed into standard deviationas using the softplus operation to ensure positive standard deviations, with 10^{-6}. All networks use ReLU activations. We train the network using Adam with standard learning rates and a batch size of 64. We pretrain the means using a cross-entropy loss to initialize the mean-field training, and train for 10 epochs.

Then we use these weights to initialize our HMC sampler. We use the NUTS implementation provided in Hamiltorch [Cobb et al., 2019]. We allow 100 steps of burn-in and then draw 1000 samples. We use a target rejection rate of 0.8, 100 leapfrog steps per sample, and a prior precision of 1.

We then calculate the empirical covariance for these samples, and set these as the covariance of a Gaussian approximation to the true posterior. We find the closes diagonal approximation to this full-covariance Gaussian. This requires a Cholesky decomposition for which we add 10^{-6} jitter to the diagonal to improve the stability of the matrix inversion.