3rd Symposium on Advances in Approximate Bayesian Inference, 2020 1–17

Expressive yet Tractable Bayesian Deep Learning via Subnetwork Inference

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

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

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

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

José Miguel Hernández-Lobato JMH233@CAM.AC.UK
University of Cambridge & Microsoft Research & The Alan Turing Institute

Abstract

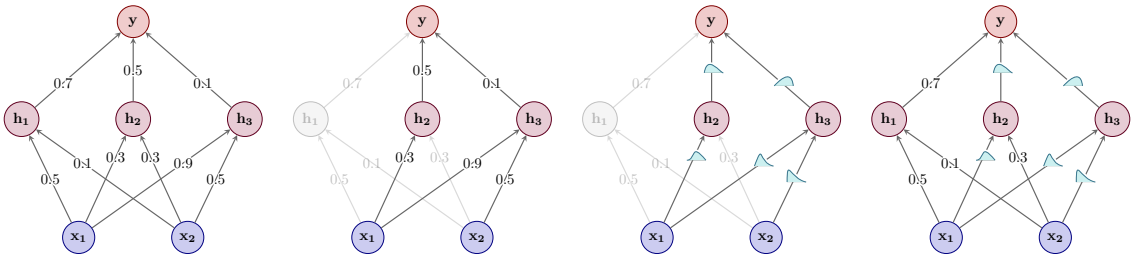
Scaling Bayesian inference to the large parameter spaces of deep neural networks requires restrictive approximations. We propose performing inference over only a small subset of the model parameters while keeping all others as point estimates. This enables us to use expressive posterior approximations that are intractable in the full model. In particular, we develop a practical and scalable Bayesian deep learning method that first trains a point estimate, and then infers a full covariance Gaussian posterior approximation over a subnetwork. We propose a subnetwork selection procedure which aims to optimally preserve posterior uncertainty. Empirical studies demonstrate the effectiveness of our approach.

1. Subnetwork Posterior Approximation

Deep neural networks (DNNs) still suffer from critical shortcomings that make them unfit for important applications. For instance, DNNs tend to be *poorly calibrated and overconfident* in their predictions, especially when there is a shift in the train and test distributions (Nguyen et al., 2015; Guo et al., 2017). To reliably inform decision making, DNNs must be able to robustly quantify the *uncertainty* in their predictions, which is particularly important in safety-critical areas such as healthcare or autonomous driving (Amodei et al., 2016).

Bayesian modeling (Ghahramani, 2015) presents a principled way to capture *model uncertainty*, i.e., uncertainty about the choice of weights $\boldsymbol{W}$ which arises due to multiple plausible explanations of the training data $\{\boldsymbol{y}, \boldsymbol{X}\}$. Here, $\boldsymbol{y}$ is the target (e.g. classification label) and $\boldsymbol{X}$ are the features. A prior distribution $p(\mathbf{W})$ is specified over the Bayesian neural network (BNN) weights. We wish to infer their full *posterior distribution* $p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \propto p(\boldsymbol{y}|\boldsymbol{X}, \mathbf{W})\, p(\mathbf{W})$. To make predictions, we then estimate the *posterior predictive* distribution that averages the network's predictions across all possible settings of the weights, weighted by their posterior probability, i.e. $p(\mathbf{y}^*|\boldsymbol{X}^*, \boldsymbol{y}, \boldsymbol{X}) = \int_{\mathbf{W}} p(\mathbf{y}^*|\boldsymbol{X}^*, \mathbf{W}) p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) d\mathbf{W}$.

* equal contribution



(*a*) Point Estimation (*b*) Subnet Selection (*c*) Bayes. Inference (*d*) Prediction

Figure 1: Schematic illustration of our proposed approach. (a) We train a neural network using standard techniques to obtain a point estimate of the weights. (b) We identify a small subnetwork within the NN. (c) We estimate a posterior distribution over the selected subnetwork using Bayesian inference techniques. (d) We make predictions using the full network comprising of both Bayesian and deterministic weights.

Unfortunately, due to the size of modern deep neural networks, it is not only intractable to infer the exact posterior distribution $p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X})$, but it is even computationally challenging to properly approximate it. As a consequence, crude posterior approximations such as complete factorization are commonly employed (Blundell et al., 2015; Gal and Ghahramani, 2016; Hernández-Lobato and Adams, 2015; Kingma et al., 2015; Khan et al., 2018; Osawa et al., 2019), i.e. $p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \approx \prod_{d=1}^{D} q(\mathrm{w}_d)$ where w_d denotes the d-th weight in the D-dimensional neural network weight vector $\mathbf{W} \in \mathbb{R}^D$ (the concatenation and flattening of all layers' weight matrices). In practice, this severely limits the expressiveness of the inferred posterior and thus deteriorates the quality of the induced predictive uncertainty estimates (Ovadia et al., 2019; Fort et al., 2019; Foong et al., 2019a,b; Ashukha et al., 2020).

In this work, we question the implicit assumption that a good posterior approximation needs to include *all* model parameters. Instead, we aim to perform inference only over a *small subset* of the model weights. This approach is well-motivated for two reasons:

1. **Overparameterization:** Maddox et al. (2020) have shown that, in the neighborhood of local optima, there are many directions that leave the NN's predictions unchanged. Moreover, NNs can be heavily pruned without sacrificing test set accuracy (Frankle and Carbin, 2019). This suggests that the majority of a NN's predictive power is isolated to a small subnetwork.

2. **Inference over submodels:** Previous work[1] has provided evidence that inference can be effective even when not done over the full parameter space. Izmailov et al. (2019) performed inference over a low-dimensional projection of the weights. Neural-linear models, which give a Bayesian treatment to only the last layer of a DNN, have empirically been shown to perform well (Riquelme et al., 2018; Ober and Rasmussen, 2019; Kristiadi et al., 2020).

We thus combine these ideas, making the following two-step approximation of the posterior:

$$p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X}) \quad \approx \quad p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*) \quad \approx \quad q(\mathbf{W}_S) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*) \,. \tag{1}$$

1. See Appendix A for a more thorough discussion of related work.

The first approximation in Eq. (1) decomposes the full neural network posterior $p(\mathbf{W}|\boldsymbol{y}, \boldsymbol{X})$ into a posterior $p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X})$ over the subnetwork $\mathbf{W}_S$ and delta functions $\delta(\mathrm{w}_r - \mathrm{w}_r^*)$ over all remaining weights $\{\mathrm{w}_r\}_r$, keeping them deterministic at fixed values $\mathrm{w}_r^* \in \mathbb{R}$. This can be viewed as *pruning the variances* of the weights $\{\mathrm{w}_r\}_r$ to zero, which is in contrast to ordinary weight pruning methods that set the *weights* $\{\mathrm{w}_r\}_r$ themselves to zero (Frankle and Carbin, 2019). The second approximation in Eq. (1) introduces the approximate distribution $q(\mathbf{W}_S)$, because exact posterior inference over the subnetwork $\mathbf{W}_S$ is still intractable. Yet, as the subnetwork is much smaller than the full network, we can afford to make $q(\mathbf{W}_S)$ expressive and able to capture rich dependencies across the weights within the subnetwork.

In this paper, we show both theoretically and empirically that the full neural network posterior can be well represented by a subnetwork's posterior as defined in Eq. (1). As a result, expressive posterior inference over a subnetwork results in better uncertainty calibration and robustness to distribution shift than crude inference over the full network.

2. Subnetwork Inference via Laplace Approximation

As a concrete instantiation of the described subnetwork inference framework, we now describe a practical and scalable Bayesian deep learning method. To this end, we use the (linearized) *Laplace approximation* (MacKay, 1992) for *post-hoc* inference of a full-covariance Gaussian posterior over a subnetwork within a pre-trained, point-estimated neural network.

Step #1: Point Estimation. The first step of the proposed procedure is to train a neural network to obtain a point estimate of the weights, denoted $\boldsymbol{W}_{MAP}$. This point estimate should respect our Bayesian model, and therefore we optimize the *maximum a-posteriori* (MAP) objective, i.e. $\boldsymbol{W}_{MAP} = \arg\max_{\boldsymbol{W}} [\log p(\mathbf{y}|\boldsymbol{X}, \boldsymbol{W}) + \log p(\boldsymbol{W})]$. This can be done using standard stochastic gradient-based optimization methods commonly-used in modern deep learning (Goodfellow et al., 2016). This step is illustrated in Fig. 1 (a).

Step #2: Subnetwork Selection. The second step is to identify a small subnetwork $\boldsymbol{W}_S$ (illustrated in Fig. 1 (b)). Ideally, we would like to find the subnetwork whose posterior is 'closest' to the full-network posterior. To this end, we first formally characterize the discrepancy between the posterior distributions over a subnetwork and the full network in terms of their 2-Wasserstein distance. We then derive a principled strategy that, under certain conditions, minimizes this discrepancy: 1) Estimate a fully-factorized posterior distribution over all weights in the full network (e.g. using a diagonal Laplace approximation) and 2) define the subnetwork to comprise of the weights with the *largest marginal variances*. The full details of this procedure are described in Appendix B. All other weights not belonging to that subnetwork are then assigned fixed values: the MAP estimates obtained in Step #1.

Step #3: Bayesian Inference. Given the subnetwork point estimate $\boldsymbol{W}_{MAP}^S$, we use the Laplace approximation to infer a full-covariance Gaussian posterior distribution over the subnetwork $\mathbf{W}_S$, i.e. $p(\mathbf{W}_S|\boldsymbol{y}, \boldsymbol{X}) \approx q(\mathbf{W}_S) = \mathcal{N}\left(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, H^{-1}\right)$. Here, the posterior covariance matrix $H^{-1} \in \mathbb{R}^{D \times D}$ corresponds to the inverse of the average Hessian of the negative log posterior, i.e. $H = N\mathbb{E}\left[-\partial^2 \log p(\mathbf{y}|\boldsymbol{X}, \boldsymbol{W})/\partial \boldsymbol{W}^2\right] + \lambda \boldsymbol{I}$, where the expectation is w.r.t. the data generating distribution, and λ is the precision of a zero-mean factorized Gaussian prior $p(\mathbf{W}) = \mathcal{N}(\mathbf{W}; \mathbf{0}, \lambda^{-1}\boldsymbol{I})$. In practice, we approximate the Hessian

H with the *generalized Gauss-Newton (GGN) matrix* $\widetilde{H}$ (Schraudolph, 2002), i.e.

$$\widetilde{H} = \sum_{n=1}^{N} \boldsymbol{J}_n^\top \boldsymbol{H}_n \boldsymbol{J}_n + \lambda \boldsymbol{I}, \quad \text{with} \quad \boldsymbol{J}_n = \frac{\partial \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{W})}{\partial \boldsymbol{W}} \quad \text{and} \quad \boldsymbol{H}_n = \frac{\partial^2 L(\boldsymbol{y}_n, \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{W}))}{\partial^2 \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{W})} \tag{2}$$

where $\boldsymbol{J}_n \in \mathbb{R}^{O \times D}$ is the Jacobian of the neural network features $\boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{W}) \in \mathbb{R}^O$ w.r.t. the weights $\boldsymbol{W}$, and $\boldsymbol{H}_n \in \mathbb{R}^{O \times O}$ is the Hessian of the loss $L(\boldsymbol{y}_n, \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{W}))$ w.r.t. the features $\boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{W})$. The GGN $\widetilde{H}$ has clear practical advantages over the Hessian H; see Martens and Sutskever (2011) and Martens (2016). Using the Laplace approximation with the GGN-approximation to the Hessian can be viewed as an implicit *local linearization* of the underlying neural network $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{W})$ at its MAP estimate $\boldsymbol{W}_{MAP}$, i.e.

$$\boldsymbol{f}_{lin}^{MAP}(\boldsymbol{x}, \boldsymbol{W}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{W}_{MAP}) + \boldsymbol{J}_{\boldsymbol{W}_{MAP}}(\boldsymbol{x})(\boldsymbol{W} - \boldsymbol{W}_{MAP}) \tag{3}$$

where $\boldsymbol{J}_{\boldsymbol{W}_{MAP}}(\boldsymbol{x}) = \partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{W}_{MAP}) / \partial \boldsymbol{W}_{MAP} \in \mathbb{R}^{O \times D}$ (Immer et al., 2020). Note that the model in Eq. (3) is *linear* in $\boldsymbol{W}$, as only the term $\boldsymbol{J}_{\boldsymbol{W}_{MAP}}(\boldsymbol{x})\boldsymbol{W}$ depends linearly on $\boldsymbol{W}$, while the other terms are constant w.r.t. $\boldsymbol{W}$ and can thus be subsumed into an additive bias term (Khan et al., 2019). The GGN approximation thus locally turns the underlying probabilistic model from a Bayesian neural network into a (generalized) linear model, with basis function expansion $\boldsymbol{J}_{\boldsymbol{W}_{MAP}}(\boldsymbol{x})$ of covariate $\boldsymbol{x}$ (Immer et al., 2020). Put differently, linearized Laplace in the neural network $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{W})$ is *equivalent* to ordinary Laplace in the linear model $\boldsymbol{f}_{lin}^{MAP}(\boldsymbol{x}, \boldsymbol{W})$ in Eq. (3), as the GGN $\widetilde{H}$ corresponding to $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{W})$ in Eq. (2) is equivalent to the Hessian H corresponding to $\boldsymbol{f}_{lin}^{MAP}(\boldsymbol{x}, \boldsymbol{W})$ in Eq. (3) (Khan et al., 2019). This is a useful property that allows us to derive a principled subnetwork selection strategy in Appendix B. This step is illustrated in Fig. 1 (c). We emphasize that this whole procedure (i.e. Steps #1-#3) is a perfectly valid mixed inference strategy, performing full Laplace inference over the selected subnetwork and MAP inference over all remaining weights.

Step #4: Prediction. Given the linearized Laplace approximation over the subnetwork $\mathbf{W}_S$, i.e. $q(\mathbf{W}_S) = \mathcal{N}(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, \widetilde{H}^{-1})$, we can then compute the posterior predictive distribution to make predictions (illustrated in Fig. 1 (d)). While, traditionally, one would compute the predictive distribution using the original Bayesian neural network likelihood, i.e. $p(\mathbf{y}|\boldsymbol{X}, \mathbf{W}) = p(\mathbf{y}|\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{W}))$, Immer et al. (2020) recently suggested that, since inference was (implicitly) done in the GGN-linearized model, it is more principled to instead predict using the linearized likelihood Eq. (3), i.e. $p(\mathbf{y}|\boldsymbol{X}, \mathbf{W}) = p(\mathbf{y}|\boldsymbol{f}_{lin}^{MAP}(\boldsymbol{x}, \boldsymbol{W}))$. This provides a formal justification for the empirical superiority of this approach observed previously (Lawrence, 2001; Foong et al., 2019b). We thus obtain the *linearized predictive distribution*

$$p(\mathbf{y}^*|\boldsymbol{X}^*, \boldsymbol{y}, \boldsymbol{X}) \approx \int_{\mathbf{W}} p(\mathbf{y}^*|\boldsymbol{f}_{lin}^{MAP}(\boldsymbol{X}^*, \boldsymbol{W}))\, \mathcal{N}(\mathbf{W}_S; \boldsymbol{W}_{MAP}^S, \widetilde{H}^{-1}) \prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)\, d\mathbf{W} \,. \tag{4}$$

There are two ways to compute Eq. (4): Firstly, via a Monte Carlo approximation $p(\mathbf{y}^*|\boldsymbol{X}^*, \boldsymbol{y}, \boldsymbol{X}) \simeq \frac{1}{M} \sum_{m=1}^{M} p(\mathbf{y}^*|\boldsymbol{f}_{lin}^{MAP}(\boldsymbol{X}^*, \widetilde{\boldsymbol{W}}_m))$ by sampling $\widetilde{\boldsymbol{W}}_m$ from $\mathcal{N}(\boldsymbol{W}_{MAP}^S, \widetilde{H}^{-1})$ and $\prod_r \delta(\mathrm{w}_r - \mathrm{w}_r^*)$, the latter of which is trivial. Secondly, due to linearity of $p(\mathbf{y}^*|\boldsymbol{f}_{lin}^{MAP}(\boldsymbol{X}^*, \boldsymbol{W}))$, there are closed-form expressions which are exact for Gaussian likelihoods (i.e. regression) and approximate for categorical ones (i.e. classification) (Bishop, 2006; Gibbs, 1998).

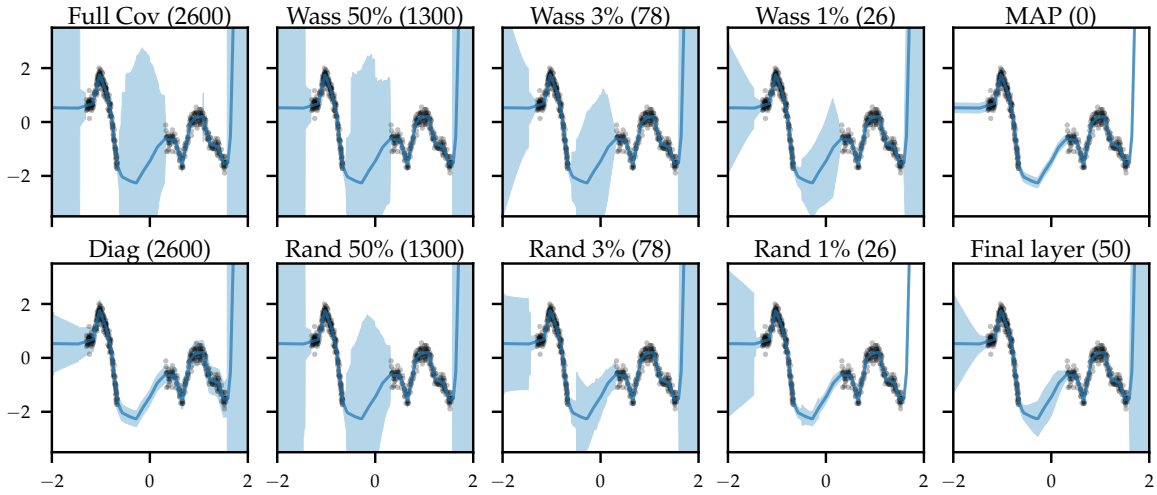


Figure 2: Predictive distributions (mean ± std) for 1D regression. In brackets: number of parameters (out of 2600) over which inference was done. Wasserstein subnetwork inference maintains richer predictive uncertainties at smaller parameter counts.

3. Empirical Analysis

We now empirically assess the effectiveness of subnetwork inference compared to point-estimated NNs and methods that do less expressive inference over the full network. We consider two tasks: 1) small-scale synthetic regression and 2) large-scale image classification. In Appendix C, we report additional empirical results on medium-scale tabular regression.

3.1. How does subnetwork inference retain posterior predictive uncertainty?

We first assess how the predictive distribution of a full-covariance Gaussian posterior over a selected subnetwork qualitatively compares to that obtained from 1) a full-covariance Gaussian over the *full* network (*Full Cov*), 2) a *factorised* Gaussian posterior over the full network (*Diag*), 3) a full-covariance Gaussian over only the (*Final layer*) of the network (Kristiadi et al., 2020), and 4) a point estimate (*MAP*). For subnetwork inference, we consider both Wasserstein (*Wass*) (as described in Appendix B) and uniform random subnetwork selection (*Rand*) to obtain subnetworks that comprise of only 50%, 3% and 1% of the model parameters. Our NN consists of 2 ReLU hidden layers with 50 hidden units each. We use a homoscedastic Gaussian likelihood with noise variance optimised via maximum likelihood. Using the linearized predictive in Eq. (4), all approaches share their predictive mean, allowing us to better compare their uncertainties. All methods share a prior precision of $\lambda = 3$. We use a synthetic 1D regression task with two separated clusters of inputs (Antorán et al., 2020), allowing us to probe for 'in-between' uncertainty (Foong et al., 2019b). Results are shown in Fig. 2. Subnetwork inference preserves more of the uncertainty of full network inference than diagonal Gaussian or final layer inference while doing inference over fewer weights. By capturing weight correlations, subnetwork inference retains uncertainty in between clusters of data. This is true for both random and Wasserstein subnetwork

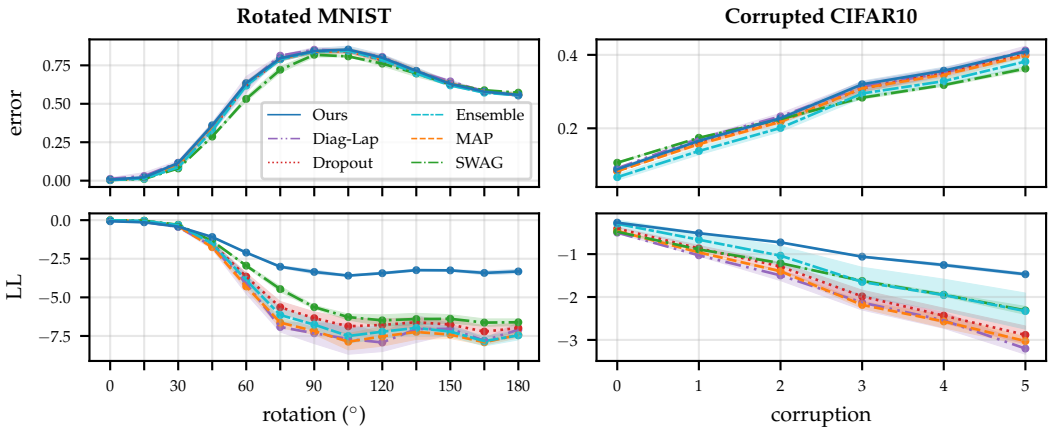


Figure 3: Results on rotated MNIST (left) and corrupted CIFAR (right), showing the error (top) and log-likelihood (bottom) (mean ± std across three seeds). Subnetwork inference retains better uncertainty calibration and robustness to distribution shift than MAP-estimated networks and other Bayesian deep learning methods.

selection. However, the latter preserves more uncertainty with smaller subnetworks. This suggests that **expressive inference over a carefully selected subnetwork retains more predictive uncertainty than crude approximations over the full network**.

3.2. How robust is our approach to distribution shift in image classification?

We assess the robustness of large CNNs with subnetwork inference to distribution shift on image classification tasks compared to the following baselines: MAP-estimated networks, Bayesian deep learning methods that do less expressive inference over the full network: MC Dropout (Gal and Ghahramani, 2016), diagonal Laplace (both of which assume factorisation of the weight posterior), and SWAG (Maddox et al., 2019) (which assumes a diagonal plus low-rank posterior). We also assess deep ensembles (Lakshminarayanan et al., 2017), which is considered state-of-the-art for uncertainty quantification in deep learning (Ovadia et al., 2019; Ashukha et al., 2020). For all approaches, we use a ResNet-18 (He et al., 2016) with 11,168,000 parameters. For our method, we retain a subnetwork with only 42,438 of them (i.e. 0.38%). See Appendices D and E for more experimental details and results, respectively.

Following Ovadia et al. (2019); Antorán et al. (2020), we perform the following two benchmark experiments: 1) We train all methods on MNIST and evaluate their predictive distributions on increasingly rotated digits (**rotated MNIST**). 2) We train all methods on CIFAR10 and evaluate on data subject to 16 different corruptions with 5 levels of intensity each (Hendrycks and Dietterich, 2019) (**corrupted CIFAR**). Results are shown in Fig. 3. While all methods perform well on the original test sets, their accuracy degrades quickly for increasing levels of rotation/corruption. Our approach matches a MAP-estimated network in terms of predictive error as local linearization makes their predictions the same. Ensembles and SWAG are the most accurate, and, out of our baselines, perform best in terms of log-likelihood. Even so, subnetwork inference differentiates itself by being the least

overconfident, outperforming all baselines in terms of log-likelihood at all rotation/corruption levels. Subnetwork inference makes accurate predictions in-distribution while assigning higher uncertainty than the baselines to out-of-distribution inputs. These results suggest that **subnetwork inference results in better uncertainty calibration and robustness to distribution shift than other popular uncertainty quantification methods**.

4. Conclusion

We develop a *practical* and *scalable* method for expressive yet tractable Bayesian deep learning. We approximate the posterior over a subnetwork within a NN while keeping all other weights deterministic. Computational cost is decoupled from network size, allowing us to *scale* expressive approximations, such as full-covariance Gaussians, to modern DNNs. Our method can be applied post-hoc to any pre-trained model, making it particularly attractive for *practical* use. Our experiments suggest that subnetwork inference 1) retains more predictive uncertainty than crude approximations over the full model, and 2) is competitive with state-of-the-art uncertainty quantification methods on real-world scale problems.

Acknowledgments

We thank Matthias Bauer and Andrew Y. K. Foong for helpful discussions and feedback. ED acknowledges funding from the EPSRC and Qualcomm. JA acknowledges support from Microsoft Research, through its PhD Scholarship Programme, and from the EPSRC. JUA acknowledges funding from the EPSRC and the Michael E. Fisher Studentship in Machine Learning. This work has been performed using resources provided by the Cambridge Tier-2 system operated by the University of Cambridge Research Computing Service (http://www.hpc.cam.ac.uk) funded by EPSRC Tier-2 capital grant EP/P020259/1.

References

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Javier Antorán, James Urquhart Allingham, and José Miguel Hernández-Lobato. Depth uncertainty in neural networks, 2020.

Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *ICLR*, 2020.

Michael Betancourt. The fundamental incompatibility of scalable hamiltonian monte carlo and naive data subsampling. volume 37 of *Proceedings of Machine Learning Research*, pages 533–540, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/betancourt15.html.

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

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. In *Proceedings of The 32nd International Conference on Machine Learning (ICML)*, pages 1613–1622, 2015.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Michael W Dusenberry, Ghassen Jerfel, Yeming Wen, Yi-an Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable bayesian neural nets with rank-1 factors. *arXiv preprint arXiv:2005.07186*, 2020.

Angelos Filos, Sebastian Farquhar, Aidan N Gomez, Tim GJ Rudner, Zachary Kenton, Lewis Smith, Milad Alizadeh, Arnoud de Kroon, and Yarin Gal. Benchmarking bayesian deep learning with diabetic retinopathy diagnosis. *Preprint*, 2019.

Andrew YK Foong, David R Burt, Yingzhen Li, and Richard E Turner. On the expressiveness of approximate inference in bayesian neural networks. *arXiv*, pages arXiv–1909, 2019a.

Andrew YK Foong, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. In-between uncertainty in bayesian neural networks. *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2019b.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow gaussian processes. In *International Conference on Learning Representations*, 2018.

Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521 (7553):452–459, 2015.

Mark N Gibbs. *Bayesian Gaussian processes for regression and classification*. PhD thesis, Citeseer, 1998.

Clark R Givens, Rae Michael Shortt, et al. A class of wasserstein metrics for probability distributions. *The Michigan Mathematical Journal*, 31(2):231–240, 1984.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

Alexander Immer, Maciej Korzepa, and Matthias Bauer. Improving predictions of bayesian neural networks via local linearization. *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2020.

Pavel Izmailov, Wesley J Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Subspace inference for bayesian deep learning. In *35th Conference on Uncertainty in Artificial Intelligence, UAI 2019*, 2019.

Mohammad Emtiyaz Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable bayesian deep learning by weight-perturbation in adam. *arXiv preprint arXiv:1806.04854*, 2018.

Mohammad Emtiyaz E Khan, Alexander Immer, Ehsan Abedi, and Maciej Korzepa. Approximate inference turns deep networks into gaussian processes. In *Advances in neural information processing systems*, pages 3094–3104, 2019.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems*, pages 2575–2583, 2015.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks. *arXiv preprint arXiv:2002.10118*, 2020.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

Neil David Lawrence. *Variational inference in probabilistic models*. PhD thesis, University of Cambridge, 2001.

Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In *International Conference on Machine Learning*, pages 1708–1716, 2016.

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13132–13143, 2019.

Wesley J Maddox, Gregory Benton, and Andrew Gordon Wilson. Rethinking parameter counting in deep models: Effective dimensionality revisited. *arXiv preprint arXiv:2003.02139*, 2020.

James Martens. *Second-order optimization for neural networks*. University of Toronto (Canada), 2016.

James Martens and Ilya Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1033–1040. Citeseer, 2011.

Alexander Graeme de Garis Matthews. *Scalable Gaussian process inference using variational methods*. PhD thesis, University of Cambridge, 2017.

Aaron Mishkin, Frederik Kunstner, Didrik Nielsen, Mark Schmidt, and Mohammad Emtiyaz Khan. Slang: Fast structured covariance approximations for bayesian deep learning with natural gradient. In *Advances in Neural Information Processing Systems*, pages 6245–6255, 2018.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do Deep Generative Models Know What They Don't Know? In *International Conference on Learning Representations (ICLR)*, 2019.

Radford M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, CAN, 1995. AAINN02676.

John Ashworth Nelder and R Jacob Baker. *Generalized Linear Models*. Wiley Online Library, 1972.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

Sebastian W Ober and Carl Edward Rasmussen. Benchmarking the neural linear model for regression. *arXiv preprint arXiv:1912.08416*, 2019.

Kazuki Osawa, Siddharth Swaroop, Anirudh Jain, Runa Eschenhagen, Richard E Turner, Rio Yokota, and Mohammad Emtiyaz Khan. Practical deep learning with Bayesian principles. *arXiv preprint arXiv:1906.02506*, 2019.

Yaniv Ovadia, Emily Fertig, Balaji Lakshminarayanan, Sebastian Nowozin, D Sculley, Joshua Dillon, Jie Ren, Zachary Nado, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13969–13980, 2019.

Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato. Bayesian batch active learning as sparse subset approximation. In *Advances in Neural Information Processing Systems*, pages 6359–6370, 2019.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. In *International Conference on Learning Representations*, 2018.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

Simone Rossi, Sebastien Marmin, and Maurizio Filippone. Walsh-hadamard variational inference for bayesian deep learning. *arXiv preprint arXiv:1905.11248*, 2019.

Nicol N Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180, 2015.

Jakub Swiatkowski, Kevin Roth, Bastiaan S Veeling, Linh Tran, Joshua V Dillon, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. The k-tied normal distribution: A compact parameterization of gaussian mean field posteriors in bayesian neural networks. *arXiv preprint arXiv:2002.02655*, 2020.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkgsACVKPH.

Appendix A. Additional Related Work

Bayesian Deep Learning. There have significant efforts to characterise the posterior distribution over NN weights $p(\boldsymbol{W}|\mathcal{D})$. Hamiltonian Monte Carlo (Neal, 1995) remains the golden standard for approximate inference in BNNs to this day. Although asymptotically unbiased, sampling based approaches are difficult to scale to the large datasets (Betancourt, 2015). As a result, approaches which find the best surrogate posterior among an approximating family (most often Gaussians) have gained popularity. The first of these was the Laplace approximation, introduced by MacKay (1992), who also proposed approximating the predictive posterior with that of the linearised model (Khan et al., 2019; Immer et al., 2020). The popularisation of larger NN models has made surrogate distributions that capture correlations between weights computationally intractable. Thus, most modern methods make use of the mean field assumption (Blundell et al., 2015; Hernández-Lobato and Adams, 2015; Gal and Ghahramani, 2016; Mishkin et al., 2018; Osawa et al., 2019). This comes at the cost of limited expressivity (Foong et al., 2019a) and empirical under-performance (Ovadia et al., 2019; Antorán et al., 2020) of uncertainty estimates. Our proposed approach recovers predictive posterior expressivity while maintaining tractability by lowering the dimensionality of the weight space considered. This allows us to scale up approximations that *do* consider weight correlations (MacKay, 1992; Louizos and Welling, 2016; Maddox et al., 2019; Ritter et al., 2018).

Neural Network Linearization. In the limit of infinite width, NNs converge to Gaussian process (GP) behaviour (Neal, 1995; Matthews, 2017; Garriga-Alonso et al., 2018). Recently, these results have been extended to finite width BNNs when the surrogate posterior is Gaussian (Khan et al., 2019). We draw upon these results to formulate a subnetwork selection strategy for BNNs. Neural linear methods perform inference over only the last layer of a NN, while keeping all other layers fixed (Snoek et al., 2015; Riquelme et al., 2018; Ovadia et al., 2019; Ober and Rasmussen, 2019; Pinsler et al., 2019; Kristiadi et al., 2020). These represent a different generalised linear model in which the basis functions are defined by the $l-1$ first layers of a NN. They can also be viewed as a special case of subnetwork inference, in which the subnetwork is simply defined to be the last NN layer.

Inference over Subspaces. The subfield of NN pruning aims to increase the computational efficiency of NNs by identifying the smallest subset of weights which are required to make accurate predictions. Approaches trade-off computational cost with compression efficiency, ranging from those that require multiple training runs (Frankle and Carbin, 2019) to those that prune before training (Wang et al., 2020). Our work differs in that it retains all NN weights but aims to find a small subset over which to perform probabilistic reasoning. More closely related work to ours is that of Izmailov et al. (2019), who propose to perform inference over a low-dimensional subspace of weights; e.g. one constructed from the principal components of the SGD trajectory. Moreover, several recent approaches use low-rank parameterizations of approximate posteriors in the context of variational inference (Rossi et al., 2019; Swiatkowski et al., 2020; Dusenberry et al., 2020). This could also be viewed as doing inference over an implicit subspace of weight space. In contrast, we propose a technique to find subsets of weights which are relevant to predictive uncertainty, i.e., we identify axis aligned subspaces.

Appendix B. Principled Subnetwork Selection for Linear(ized) Models

We next analyze the subnetwork inference procedure described in Section 2 for the case of a *generalized linear model* (GLM) (Nelder and Baker, 1972), which models the expected response y_n given the basis function expansion of the covariates $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\boldsymbol{x}_n)$ as

$$\mathbb{E}[y_n|\boldsymbol{\phi}_n] = g^{-1}(\mathbf{w}^T\boldsymbol{\phi}_n). \tag{5}$$

Here, $\mathbf{w} \in \mathbb{R}^D$ is the vector of model parameters (which subsumes a scalar bias β_0 for notational convenience) and $g^{-1}(\cdot)$ denotes a *link function* such that $g^{-1} : \mathbb{R} \mapsto \mu_{y|\boldsymbol{\phi}}$. In particular, we consider a *Bayesian* GLM, by specifying a prior distribution $p(\mathbf{w})$ over model parameters and aiming to infer the posterior distribution $p(\mathbf{w}|\mathbf{y}, \boldsymbol{\Phi}) \propto p(\mathbf{y}|\boldsymbol{\Phi}, \mathbf{w})p(\mathbf{w})$, where $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, ... \boldsymbol{\phi}_N]^T$.

1. **Point Estimation.** Obtain the MAP estimate, $\mathbf{w}_{MAP} = \arg\max_{\mathbf{w}} \log p(\boldsymbol{y}|\boldsymbol{\Phi}, \mathbf{w}) + \log p(\mathbf{w})$. For commonly-used link functions (e.g. the identity in case of a Gaussian likelihood for regression, or the sigmoid/softmax function in case of a categorical likelihood for classification) and commonly-used priors (e.g. a Gaussian), the log-posterior $\propto \log p(\boldsymbol{y}|\boldsymbol{\Phi}, \mathbf{w}) + \log p(\mathbf{w})$ is concave. This allows for simple gradient-based MAP optimisation. It also makes a full-covariance Gaussian, estimated via Laplace, a faithful approximation to the true, uni-modal posterior, i.e.

$$p(\mathbf{w}|\mathbf{y}, \boldsymbol{\Phi}) \approx \widetilde{p}(\mathbf{w}|\mathbf{y}, \boldsymbol{\Phi}) = \mathcal{N}(\mathbf{w}; \mathbf{w}_{MAP}, H^{-1}) \tag{6}$$

   where H is the Hessian defined in Section 2. Note that for the GLM we consider, the Hessian H is equivalent to the GGN $\widetilde{H}$ defined in Eq. (2), meaning that an *ordinary* Laplace approximation is equivalent to a *linearized* Laplace approximation (Martens, 2016). For the case of an identity link function (i.e. a Gaussian likelihood with noise variance σ_0^2) and a Gaussian prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_0^{-1})$, the MAP estimate even has a closed-form expression, $\mathbf{w}_{MAP} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \sigma_0^2\boldsymbol{\Lambda}_0)^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}$. Here, the Laplace approximation in Eq. (6) *exactly corresponds to the true posterior*, i.e. $\widetilde{p}(\mathbf{w}|\mathbf{y}, \boldsymbol{\Phi}) = p(\mathbf{w}|\mathbf{y}, \boldsymbol{\Phi})$. We will thus refer to the posterior $\widetilde{p}(\mathbf{w}|\mathbf{y}, \boldsymbol{\Phi})$ in Eq. (6) as the *full posterior*.

2. **Subnetwork Selection.** Select a subset of S model weights via a method of choice, yielding a binary vector $\boldsymbol{m} \in \mathbb{R}^D$ where $m_d = 1$ if the d-th weight is part of the subset, and $m_d = 0$ otherwise. For convenience, we define the binary mask matrix $\boldsymbol{M}_S = \boldsymbol{m}\boldsymbol{m}^\top \in \mathbb{R}^{D\times D}$ which contains 1s in the rows/columns corresponding to the S subnetwork weights[2], and 0s otherwise.

3. **Bayesian Inference.** Compute the posterior over the subnetwork via a Laplace approximation:

$$p_S(\mathbf{w}|\boldsymbol{y}, \boldsymbol{\Phi}) = \mathcal{N}(\mathbf{w}; \mathbf{w}_{MAP}, \boldsymbol{M}_S \odot H^{-1}) \ . \tag{7}$$

   Firstly, note that the *mean* of the subnetwork posterior in Eq. (7) is the MAP estimate $\mathbf{w}_{MAP}$ and thus equal to the mean of the full posterior $\widetilde{p}(\mathbf{w}|\mathbf{y}, \boldsymbol{\Phi})$ in Eq. (6). Secondly, note that the *covariance matrix* of the subnetwork posterior in Eq. (7) is the element-wise product $\boldsymbol{M}_S \odot H^{-1}$, which masks the (co-)variances of all weights *not* belonging to the

2. For consistency, we will keep referring to the S selected linear model weights as a "subnetwork".

subnetwork to zero, effectively making them deterministic. More precisely, the subnetwork covariance matrix, $\boldsymbol{M}_S \odot H^{-1}$, is a $D \times D$ matrix that is equal to the full posterior covariance matrix H^{-1} in the rows/columns of the S weights in the subnetwork, and zero in the rows/columns of all other $D - S$ weights.

We consider what we term the *posterior gap*—the Wasserstein distance[3] (in particular the squared 2-Wasserstein distance) between the posterior distribution over the full network and the posterior distribution over the subnetwork.

Proposition 1 (Posterior Gap) *For a subnetwork of size $S < D$, the Wasserstein gap between the full posterior $\widetilde{p}(\mathbf{w}|\boldsymbol{y}, \boldsymbol{\Phi})$ in Eq. (6) and the subnetwork posterior $p_{\mathrm{S}}(\mathbf{w}|\boldsymbol{y}, \boldsymbol{\Phi})$ in Eq. (7) is:*

$$W\left[\widetilde{p}(\mathbf{w}|\boldsymbol{y}, \boldsymbol{\Phi}) \;||\; p_{\mathrm{S}}(\mathbf{w}|\boldsymbol{y}, \boldsymbol{\Phi})\right] = \sum_{d=1}^{D} (1 + m_{dd})\, \sigma_d^2 - \mathrm{trace}(2(H^{-1}(\boldsymbol{M}_S \odot H^{-1}))^{1/2}) \;. \quad (8)$$

Proof Note that the posterior distributions $\widetilde{p}(\mathbf{w}|\boldsymbol{y}, \boldsymbol{\Phi})$ and $p_S(\mathbf{w}|\boldsymbol{y}, \boldsymbol{\Phi})$ are both Gaussian. We thus consider the squared 2-Wasserstein distance between two Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, which has the following closed-form expression (Givens et al., 1984)[4]:

$$W\left[\mathcal{N}\left(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right) \;||\; \mathcal{N}\left(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2\right)\right] = \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|_2^2 + \mathrm{trace}\left(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2 - 2\left(\boldsymbol{\Sigma}_1 \boldsymbol{\Sigma}_2\right)^{1/2}\right) \;. \quad (9)$$

Plugging in $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \mathbf{w}_{MAP}$, $\boldsymbol{\Sigma}_1 = H^{-1}$ and $\boldsymbol{\Sigma}_2 = \boldsymbol{M}_S \odot H^{-1}$, we obtain

$$\begin{aligned}
&W\left[\widetilde{p}(\mathbf{w}|\boldsymbol{y}, \boldsymbol{X}) \;||\; p_{\mathrm{S}}(\mathbf{w}|\boldsymbol{y}, \boldsymbol{X})\right] \\
&\quad = W\left[\mathcal{N}(\mathbf{w}_{MAP}, H^{-1}) \;||\; \mathcal{N}(\mathbf{w}_{MAP}, \boldsymbol{M}_S \odot H^{-1})\right] \\
&\quad = \cancel{\|\mathbf{w}_{MAP} - \mathbf{w}_{MAP}\|_2^2} + \mathrm{trace}\left(H^{-1} + (\boldsymbol{M}_S \odot H^{-1}) - 2\left(H^{-1}(\boldsymbol{M}_S \odot H^{-1})\right)^{1/2}\right) \\
&\quad = \mathrm{trace}\left((\mathbf{1} + \boldsymbol{M}_S) \odot H^{-1}\right) - \mathrm{trace}\left(2\left(H^{-1}\left(\boldsymbol{M}_S \odot H^{-1}\right)\right)^{1/2}\right) \\
&\quad = \sum_{d=1}^{D} (1 + m_{dd})\, \sigma_d^2 - \mathrm{trace}\left(2\left(H^{-1}\left(\boldsymbol{M}_S \odot H^{-1}\right)\right)^{1/2}\right)
\end{aligned}$$

■

The optimal subnetwork should then minimize the posterior gap in Eq. (8). However, for full covariance matrices H^{-1} and a large number of weights D, this will generally be infeasible as Eq. (8) depends on *all* entries of the $D \times D$-matrix H^{-1}, which is intractable to compute/store. To derive a practical subnetwork selection strategy, we assume the covariance matrix to be diagonal.

Corollary 2 (Optimality of Maximum Variance Subnetwork Selection under Decorrelation) *For a generalized linear model with posterior covariance matrix $H^{-1} = diag(\sigma_1^2, \ldots, \sigma_D^2)$, the optimal subnetwork under the Wasserstein gap is comprised of the S weights with the largest variances σ_d^2.*

3. We use the Wasserstein distance instead of the more common Kullback–Leibler divergence because the Wasserstein is well-defined for degenerate distributions and is an actual distance metric (i.e. symmetric).
4. This also holds for our case of a degenerate Gaussian with singular covariance matrix (Givens et al., 1984).

Proof For $H^{-1} = \text{diag}(\sigma_1^2, \dots, \sigma_D^2)$, the Wasserstein posterior gap in Eq. (8) simplifies to

$$W\left[\widetilde{p}(\mathbf{w}|\boldsymbol{y}, \boldsymbol{\Phi}) \;||\; p_{\text{S}}(\mathbf{w}|\boldsymbol{y}, \boldsymbol{\Phi})\right] = \sum_{d=1}^{D} \left((1 + m_{dd})\,\sigma_d^2 - 2m_{dd}\sigma_d^2\right) \;. \tag{10}$$

The optimal subnetwork selection strategy amounts to choosing the binary vector $\boldsymbol{m} = [m_{dd}]_{d=1}^{D}$ with $\sum_{d=1}^{D} m_d = S$ (i.e., we select S out of D parameters) s.t. the posterior gap in Eq. (10) is *minimized*. Observing that the contribution of the d-th parameter to the posterior gap is $(1+1)\sigma_d^2 - 1 \times 2\sigma_d^2 = 0$ if it is selected (i.e. if $m_{dd} = 1$), and $(1+0)\sigma_d^2 - 0 \times 2\sigma_d^2 = \sigma_d^2$ if it is *not* selected (i.e. if $m_{dd} = 0$), we see that the optimal subnetwork comprises of the S weights with the *largest* variances σ_d^2. ■

Finally, since a GGN-linearized neural network, as in Eq. (3), corresponds to a GLM with basis expansion $\boldsymbol{\phi}_n = \boldsymbol{J}_{\boldsymbol{W}_{MAP}}(\boldsymbol{x}_n) = \partial \boldsymbol{f}(\boldsymbol{x}_n, \boldsymbol{W}_{MAP})/\partial \boldsymbol{W}_{MAP}$ (see Step #3 in Section 2), Theorem 2 implies that the optimal subnetwork comprises of the weights with the largest variances (assuming decorrelation). In practice, even just computing the diagonal of the covariance matrix is challenging, so we use a diagonal Laplace approximation which instead computes the inverse of the diagonal of the GGN (see e.g. Ritter et al. (2018)). We will also refer to the procedure described in Theorem 2 as *Wasserstein pruning* in the experiments in Section 3, as it effectively *prunes* the variances of all weights that do *not* belong to the subnetwork found via Wasserstein posterior gap minimization. Finally, note that we only have to make the decorrelation assumption for the purposes of subnetwork selection – when doing posterior inference over the selected subnetwork, *we estimate a full covariance matrix* for maximal expressiveness, as described in Step #3 in Section 2.

Appendix C. Subnetwork inference in large models vs full inference over small models

We study the following natural question: "Why should one use subnetwork inference in a large model when one can just perform full network inference over a smaller model?" We explore this by considering 4 fully connected models of increasing size. These have numbers of hidden layers h_d=$\{1, 2\}$ and hidden layer widths w_d=$\{50, 100\}$. For a dataset with input dimension i_d, the number of weights is given by D=$(i_d$+1$)w_d$+$(h_d$−1$)w_d^2$. Our 2 hidden layer, 100 hidden unit models have a weight count of the order 10^4. Full covariance inference in these models borders the limit of computational tractability on commercial hardware. We first obtain a MAP estimate of each model's weights and our homoscedastic likelihood function's noise variance. We then perform full network GGN Laplace inference for each model. We also use our proposed Wassertein rule to prune every network's weight variances such that the number of variances that remain matches the size of every smaller network under consideration. In all cases, we employ the linearized predictive in Eq. (3). Consequently, networks with the same number of weights make the same mean predictions. Increasing the number of weight variances considered will thus only increase predictive uncertainty.

We employ 3 UCI (Dua and Graff, 2017) datasets of increasing size (input dimensionality, n. points): wine (11, 1439), kin8nm (8, 7373) and protein (9, 41157). We consider their

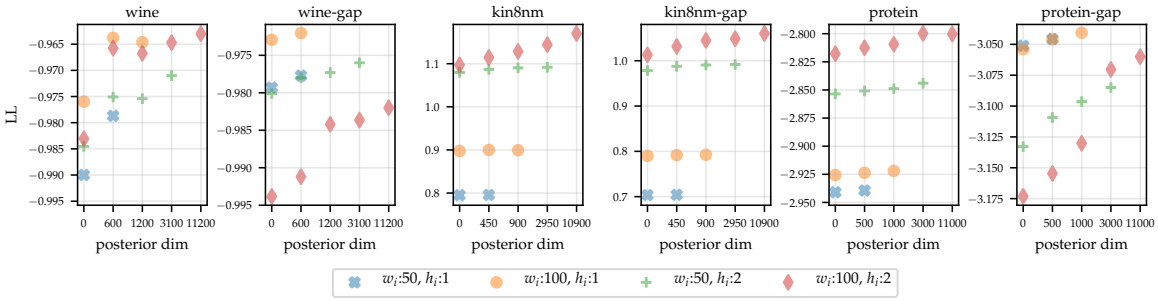


Figure 4: Mean test log-likelihood values obtained on UCI datasets across all splits. Different markers indicate models with different numbers of weights. The horizontal axis indicates the number of weights over which full covariance inference is performed. 0 corresponds to MAP parameter estimation, and the rightmost setting for each marker corresponds to full network inference.

standard train-test splits (Hernández-Lobato and Adams, 2015) and their gap variants (Foong et al., 2019b), designed to test for out-of-distribution uncertainty. For each split, we set aside 15% of the train data as a validation set. We use these for early stopping when finding MAP estimates and for selecting the weights' prior precision. We keep other hyperparameters fixed across all models and datasets. Results are in Fig. 4.

We present mean test log-likelihood (LL) values, as these take into account both accuracy and uncertainty. Larger models tend to perform better when doing MAP inference, with wine-gap and protein-gap being exceptions. We also find larger models improve over their respective MAP LLs more than small ones when performing approximate inference over the same numbers of weights. We conjecture this is due to an abundance of degenerate directions (weights) in the weight posterior of all models (Maddox et al., 2020). Full network inference in small models captures information about both useful and non-useful weights. In larger models, our pruning strategy allows us to dedicate a larger proportion of our resources to modelling informative weight variances and covariances. In 3 out of 6 datasets, we find abrupt increases in LL as we increase the number of weights over which we perform inference, followed by a plateau. Such plateaus might be explained by all of the most informative weight variances having already been accounted for. These results suggest that **subnetwork inference in a large model is superior to full network inference in a small one.**

Appendix D. Details on Image Classification Experiments

We use deep ensembles of 5 DNNs, as suggested by (Ovadia et al., 2019), and 16 samples for MC Dropout, diagonal Laplace and SWAG. We use a Dropout probability of 0.1 and a prior precision of $\lambda = 40,000$ for diagonal Laplace, found via grid search. For subnetwork inference, we compute the linearized predictive distribution in Eq. (4) via the closed-form approximation for integrals between Gaussians and multi-class cross-entropy likelihoods described in (Gibbs, 1998). We use Wasserstein pruning to retain only 0.38% of the weights, yielding a subnetwork with only 42,438 weights. This is the largest subnetwork for which we

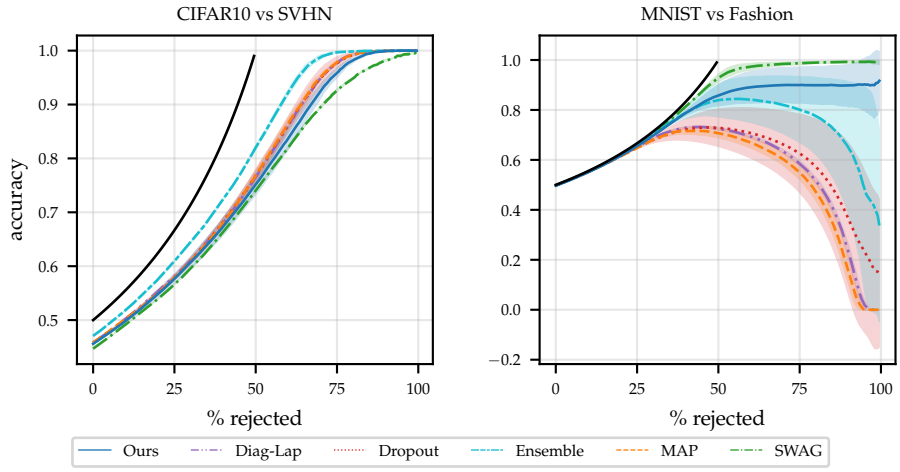


Figure 5: Rejection-classification plots. We simulate a realistic OOD rejection scenario (Filos et al., 2019) by jointly evaluating our models on an in-distribution and an OOD test set. We allow our methods to reject increasing proportions of the data based on predictive entropy before classifying the rest. All predictions on OOD samples are treated as incorrect. Following (Nalisnick et al., 2019), we use CIFAR10 vs SVHN and MNIST vs FashionMNIST as in- and out-of-distribution datasets, respectively. Note that the SVHN test set is randomly sub-sampled down to a size of 10,000.

can tractably compute a full covariance matrix. Its size is $42,438^2 \times 4$ Bytes ≈ 7.2 GB. We use a prior precision of $\lambda = 500$, found via grid search.

Appendix E. Additional Image Classification Results

Source	Target	Diag-Lap	Dropout	Ensemble	Ours	MAP	SWAG
CIFAR10	SVHN	0.86±0.02	0.86±0.01	0.91±0.00	0.85±0.03	0.86±0.02	0.83±0.00
MNIST	Fashion	0.75±0.01	0.76±0.09	0.88±0.08	0.92±0.05	0.72±0.03	0.97±0.01

Table 1: AUC-ROC scores for out-of-distribution detection, using CIFAR10 vs SVHN and MNIST vs FashionMNIST as in- (source) and out-of-distribution (target) datasets, respectively (Nalisnick et al., 2019).

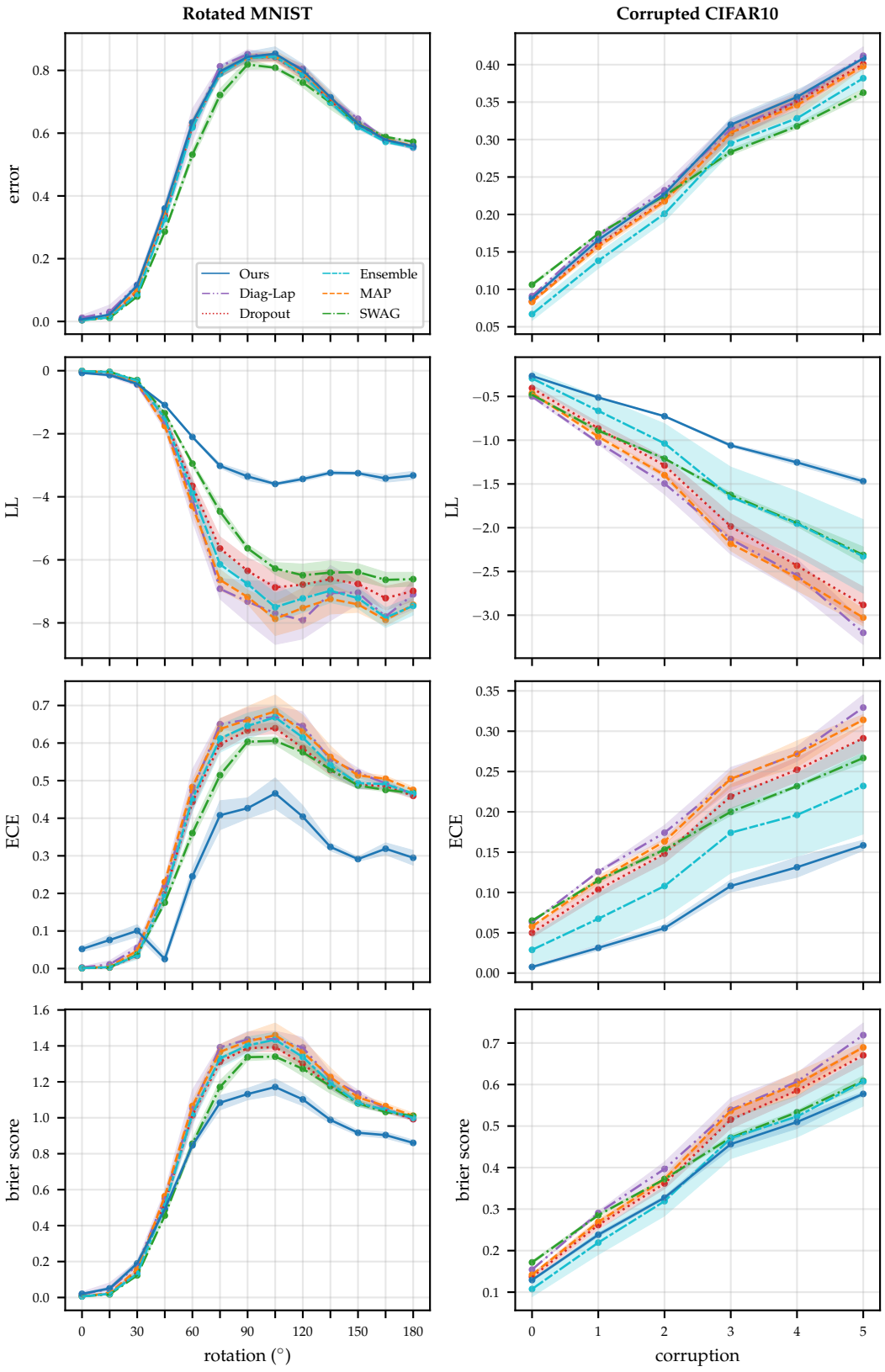


Figure 6: Full MNIST rotation and CIFAR10 corruption results, reporting predictive error, log-likelihood (LL), expected calibration error (ECE) and brier score, respectively (from top to bottom).

Method	LL	error	ECE	Brier score	dataset
Diag-Laplace	$-0.50_{\pm 0.02}$	$0.09_{\pm 0.00}$	$0.06_{\pm 0.00}$	$0.15_{\pm 0.00}$	CIFAR10
Dropout	$-0.40_{\pm 0.04}$	$0.08_{\pm 0.00}$	$0.05_{\pm 0.01}$	$0.14_{\pm 0.01}$	CIFAR10
Ensemble	$-0.29_{\pm 0.09}$	$0.07_{\pm 0.01}$	$0.03_{\pm 0.02}$	$0.11_{\pm 0.02}$	CIFAR10
Ours	$-0.27_{\pm 0.00}$	$0.09_{\pm 0.00}$	$0.01_{\pm 0.00}$	$0.13_{\pm 0.00}$	CIFAR10
MAP	$-0.46_{\pm 0.02}$	$0.08_{\pm 0.00}$	$0.06_{\pm 0.00}$	$0.14_{\pm 0.00}$	CIFAR10
SWAG	$-0.48_{\pm 0.01}$	$0.11_{\pm 0.00}$	$0.07_{\pm 0.00}$	$0.17_{\pm 0.00}$	CIFAR10
Diag-Laplace	$-0.04_{\pm 0.03}$	$0.01_{\pm 0.01}$	$0.00_{\pm 0.00}$	$0.02_{\pm 0.01}$	MNIST
Dropout	$-0.01_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.01_{\pm 0.00}$	MNIST
Ensemble	$-0.01_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.01_{\pm 0.00}$	MNIST
Ours	$-0.07_{\pm 0.01}$	$0.01_{\pm 0.00}$	$0.05_{\pm 0.01}$	$0.02_{\pm 0.00}$	MNIST
MAP	$-0.01_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.01_{\pm 0.00}$	MNIST
SWAG	$-0.01_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.01_{\pm 0.00}$	MNIST

Table 2: Results for CIFAR10 and MNIST without corruptions or rotations.