
Improving predictions of Bayesian neural networks via local linearization

Alexander Immer^{*1} Maciej Korzepa^{*2} Matthias Bauer³

Abstract

In this paper we argue that in Bayesian deep learning, the frequently utilized generalized Gauss-Newton (GGN) approximation should be understood as a modification of the underlying probabilistic model and should be considered separately from further approximate inference techniques. Applying the GGN approximation turns a BNN into a locally linearized generalized linear model or, equivalently, a Gaussian process. Because we then use this linearized model for inference, we should also predict using this modified likelihood rather than the original BNN likelihood. This formulation extends previous results to general likelihoods and alleviates underfitting behaviour observed e.g. by Ritter et al. (2018). We demonstrate our approach on several UCI classification datasets as well as CIFAR10.

1. Introduction

Inference in Bayesian neural networks (BNNs) usually requires posterior approximations due to intractable integrals and high computational cost. Given such an approximate posterior, one can make predictions by combining it with the *original* Bayesian neural network likelihood.

Recently, Foong et al. (2019) showed empirically that predictions using a “linearized Laplace” predictive approximation can match or outperform other approximate inference approaches, such as mean field variational inference (MFVI) in the *original* BNN model (Blundell et al., 2015) and provide better “in-between” uncertainties. Here we explain that their approach relies on an implicit change in probabilistic model due to the *generalized Gauss-Newton* (GGN) approximation. The GGN is often jointly applied with approximate inference in BNNs using the Laplace (Ritter et al., 2018; Foresee & Hagan, 1997; Foong et al., 2019) or variational approximation (Khan et al., 2018; Zhang et al., 2018).

^{*}Equal contribution ¹ETH Zürich, CH and MPI-IS, DE ²DTU, Copenhagen, DK ³DeepMind, London, UK. ^{1,2,3}Work partly done during an internship at RIKEN AIP, Japan. Correspondence to: Matthias Bauer <msbauer@google.com>.

Presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning. Copyright 2020 by the author(s).

We argue that the GGN and approximate inference should really be considered separately: (1) the GGN locally linearizes the underlying probabilistic model in the parameters and gives rise to a generalized linear model (GLM); (2) approximate inference enables posterior inference in this linearized GLM. Because we have done inference in a *modified* probabilistic model (the GLM), we should also predict with this modified model. We call the resulting predictive the “GLM predictive” in contrast to the “BNN predictive” that uses the original BNN likelihood, see Fig. 1.

Our formulation generalizes previous results by Khan et al. (2019) and Foong et al. (2019) to non-Gaussian likelihoods and explains why the GLM predictive works well compared to the BNN predictive. The BNN predictive with Laplace approximation can show underfitting (Lawrence, 2001) especially severely when combined with the GGN (Ritter et al., 2018). Our experiments show that the GLM predictive alleviates these underfitting issues and consistently outperforms the BNN predictive; it is on par or better than the neural network MAP or MFVI. Further, the GLM in weight space can be viewed as an equivalent Gaussian process (GP) in function space, which enables different inference algorithms.

BNN predictive (Eq. (3))	GLM predictive (Eq. (8))
$p_{\text{BNN}}(\mathbf{y} \mathbf{x}, \mathcal{D}) = \int q(\boldsymbol{\theta})p(\mathbf{y} \mathbf{f}(\mathbf{x}, \boldsymbol{\theta})) d\boldsymbol{\theta}$	$p_{\text{GLM}}(\mathbf{y} \mathbf{x}, \mathcal{D}) = \int q(\boldsymbol{\theta})p(\mathbf{y} \mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}, \boldsymbol{\theta})) d\boldsymbol{\theta}$
	$\mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{f}(\mathbf{x}, \boldsymbol{\theta}^*) + \nabla_{\boldsymbol{\theta}}\mathbf{f}(\mathbf{x}, \boldsymbol{\theta}) _{\boldsymbol{\theta}=\boldsymbol{\theta}^*}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)$

Figure 1. The generalized Gauss Newton approximation (GGN) turns a Bayesian neural network (BNN) into a generalized linear model (GLM) with same likelihood distribution, but network function $\mathbf{f}(\mathbf{x}, \boldsymbol{\theta})$ linearized around $\boldsymbol{\theta}^*$. When using GGN, we should use the GLM predictive and not the BNN predictive. $q(\boldsymbol{\theta})$ is an approximate posterior and $\boldsymbol{\theta}^*$ is found by MAP estimate, Eq. (1).

2. Methods

2.1. Background

We consider a supervised learning task with inputs $\mathbf{x}_n \in \mathbb{R}^D$ and outputs $\mathbf{y}_n \in \mathbb{R}^C$ (regression) or $\mathbf{y}_n \in \{0, 1\}^C$ (classification), $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_n^N$, and a probabilistic model with likelihood parameters $\boldsymbol{\theta}$, $p(\mathcal{D}|\boldsymbol{\theta}) = \prod_n p(\mathbf{y}_n|\mathbf{f}(\mathbf{x}_n, \boldsymbol{\theta}))$. The features $\mathbf{f}(\mathbf{x}, \boldsymbol{\theta})$ are mapped to the outputs using an inverse link function \mathbf{g}^{-1} , $\mathbb{E}[\mathbf{y}] = \mathbf{g}^{-1}(\mathbf{f}(\mathbf{x}, \boldsymbol{\theta}))$.

In Bayesian DL we impose a prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{m}_0, \mathbf{S}_0)$ and aim to compute the posterior $p(\boldsymbol{\theta}|\mathcal{D})$. This posterior inference requires computation of a high-dimensional integral, the model evidence or marginal likelihood $p(\mathbf{y}|\mathbf{x}) = \int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$. Exact inference is usually not possible and approximation techniques are used: $q(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D})$. Many practical approaches compute the maximum a posteriori (MAP) solution given by $\boldsymbol{\theta}_{\text{MAP}} = \arg \max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta})$ and return a point estimate or a distribution $q(\boldsymbol{\theta})$ around $\boldsymbol{\theta}_{\text{MAP}}$,

$$\ell(\boldsymbol{\theta}) = \sum_{n=1}^N \log p(\mathbf{y}_i|\mathbf{f}(\mathbf{x}_i, \boldsymbol{\theta})) + \log p(\boldsymbol{\theta}). \quad (1)$$

Popular in recent years, **mean-field variational inference** (MFVI) approximates the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ by a factorized variational distribution $q(\boldsymbol{\theta})$ optimized using an evidence lower bound (ELBO) to the marginal likelihood.

To obtain the **BNN predictive** distribution, we integrate the (approximate) posterior $q(\boldsymbol{\theta})$ against the BNN likelihood:

$$p_{\text{BNN}}(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \mathbb{E}_{q(\boldsymbol{\theta})} [p(\mathbf{y}|\mathbf{f}(\mathbf{x}, \boldsymbol{\theta}))] \quad (2)$$

$$\approx \frac{1}{M} \sum_m p(\mathbf{y}|\mathbf{f}(\mathbf{x}, \boldsymbol{\theta}_m)), \quad \boldsymbol{\theta}_m \sim q(\boldsymbol{\theta}) \quad (3)$$

Curvature (Hessian) information can be used for optimization or approximate inference. We express the involved Jacobian and Hessian of the log likelihood through the *Jacobian* $\mathcal{J} \in \mathbb{R}^{C \times P}$ and *Hessian* $\mathcal{H} \in \mathbb{R}^{C \times P \times P}$ of the feature extractor \mathbf{f} , $[\mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x})]_{ci} = \frac{\partial f_c(\mathbf{x}, \boldsymbol{\theta})}{\partial \theta_i}$ and $[\mathcal{H}_{\boldsymbol{\theta}}(\mathbf{x})]_{cij} = \frac{\partial^2 f_c(\mathbf{x}, \boldsymbol{\theta})}{\partial \theta_i \partial \theta_j}$:

$$\nabla_{\boldsymbol{\theta}} \log p(\mathbf{y}|\mathbf{f}(\mathbf{x}, \boldsymbol{\theta})) = \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x})^T \mathbf{r}(\mathbf{y}; \mathbf{f}) \quad (4)$$

$$\begin{aligned} \nabla_{\boldsymbol{\theta}\boldsymbol{\theta}}^2 \log p(\mathbf{y}|\mathbf{f}(\mathbf{x}, \boldsymbol{\theta})) &= \mathcal{H}_{\boldsymbol{\theta}}(\mathbf{x})^T \mathbf{r}(\mathbf{y}; \mathbf{f}) \\ &\quad - \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x})^T \boldsymbol{\Lambda}(\mathbf{y}; \mathbf{f}) \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x}), \end{aligned} \quad (5)$$

where $\mathbf{r}(\mathbf{y}; \mathbf{f}) = \nabla_{\mathbf{f}} \log p(\mathbf{y}|\mathbf{f})$ can be interpreted as a residual, and $\boldsymbol{\Lambda}(\mathbf{y}; \mathbf{f}) = -\nabla_{\mathbf{f}\mathbf{f}}^2 \log p(\mathbf{y}|\mathbf{f})$.

2.2. Generalized Gauss-Newton turns Bayesian neural nets into generalized linear models

The network Hessian $\mathcal{H}_{\boldsymbol{\theta}}(\mathbf{x})$ in Eq. (5) is infeasible to compute, such that many approaches employ the *generalized Gauss-Newton* (GGN) approximation, which drops this term (Martens, 2014). The GGN can be viewed as an approximation that becomes exact if the network is a perfect predictor, $\mathbf{r}(\mathbf{y}; \mathbf{f}(\mathbf{x}, \boldsymbol{\theta})) = 0 \forall (\mathbf{x}, \mathbf{y})$. However, this is neither desired (it indicates overfitting) nor realistic.

Alternatively, the GGN can be interpreted as a *local linearization* of the network function $\mathbf{f}(\mathbf{x}, \boldsymbol{\theta})$: $\mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{f}(\mathbf{x}, \boldsymbol{\theta}^*) + \mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x})(\boldsymbol{\theta} - \boldsymbol{\theta}^*)$ at a parameter setting $\boldsymbol{\theta}^*$ (Bottou et al., 2018), (\rightarrow in Fig. A1). This linearization reduces the BNN to a Bayesian generalized linear model (GLM) with log joint distribution

$$\sum_{n=1}^N \log p(\mathbf{y}_n|\mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}_n, \boldsymbol{\theta})) + \log p(\boldsymbol{\theta}), \quad (6)$$

where $\mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}, \boldsymbol{\theta})$ is linear in $\boldsymbol{\theta}$ but not in the input data \mathbf{x} . Applying the GGN approximation turns the underlying probabilistic model locally from a BNN into a GLM.

In practice, the linearization point $\boldsymbol{\theta}^*$ is given by a MAP estimate found by regular (S)GD of Eq. (1).

2.3. Approximate inference in the GLM

For general likelihoods, inference in $\boldsymbol{\theta}$ is still intractable, and we have to resort to approximate inference. Here we focus on the Laplace approximation and variational inference, both of which lead to a Gaussian posterior approximation $q(\boldsymbol{\theta})$ to $p(\boldsymbol{\theta}|\mathcal{D})$ (\rightarrow in Fig. A1), and we consider full covariance and diagonal approximations. Both approximations are computed iteratively for general likelihoods, see e.g. Bishop (2006, Chapter 4), whereas for Gaussian likelihoods they can be evaluated in a single step.

When the GGN and Laplace approximation are applied jointly one assumes that (S)GD optimization of Eq. (1) finds $\boldsymbol{\theta}^* \approx \boldsymbol{\theta}_{\text{MAP}}$ and constructs a posterior approximation with mode $\boldsymbol{\theta}^*$ (Ritter et al., 2018; Foong et al., 2019). We found that approximate inference in the GLM model at $\boldsymbol{\theta}^*$ was able to improve on it using several update steps; i.e. the mode of $q(\boldsymbol{\theta})$ can be different from $\boldsymbol{\theta}^*$. The GLM objective is convex and therefore easier to optimize and guarantees convergence.

2.4. The GLM predictive distribution

To make predictions, we combine the approximate posterior with the likelihood. *Because we have done inference in the GGN-linearized model, we should predict using this modified model and not the original BNN likelihood:*

$$p_{\text{GLM}}(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \mathbb{E}_{q(\boldsymbol{\theta})} [p(\mathbf{y}|\mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}, \boldsymbol{\theta}))] \quad (7)$$

$$\approx \frac{1}{M} \sum_m p(\mathbf{y}|\mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}, \boldsymbol{\theta}_m)), \quad \boldsymbol{\theta}_m \sim q(\boldsymbol{\theta}). \quad (8)$$

We stress that the GLM predictive in Eq. (8) typically uses the same approximate posterior as the BNN predictive, Eq. (3), but locally linearized features in the likelihood.

2.5. Gaussian process formulation of the GLM

A Bayesian GLM in weight space is equivalent to a (generalized) Gaussian process (GGP) in function space with a particular kernel (\leftrightarrow in Fig. A1) (Rasmussen & Williams, 2006). The corresponding log joint is given by $\sum_{n=1}^N \log p(\mathbf{y}_n|\mathbf{f}_n) + \log p(\mathbf{f})$, with GP prior $p(\mathbf{f})$ mean and covariance function:

$$\begin{aligned} \mathbf{m}(\mathbf{x}) &= \mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}; \mathbf{m}_0) \\ \mathbf{k}(\mathbf{x}, \mathbf{x}') &= \mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x})\mathbf{S}_0\mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x}')^T, \end{aligned} \quad (9)$$

where \mathbf{m}_0 and \mathbf{S}_0 are the prior parameters of $p(\boldsymbol{\theta})$. Similar to the GLM, we can derive a GGP predictive and perform ap-

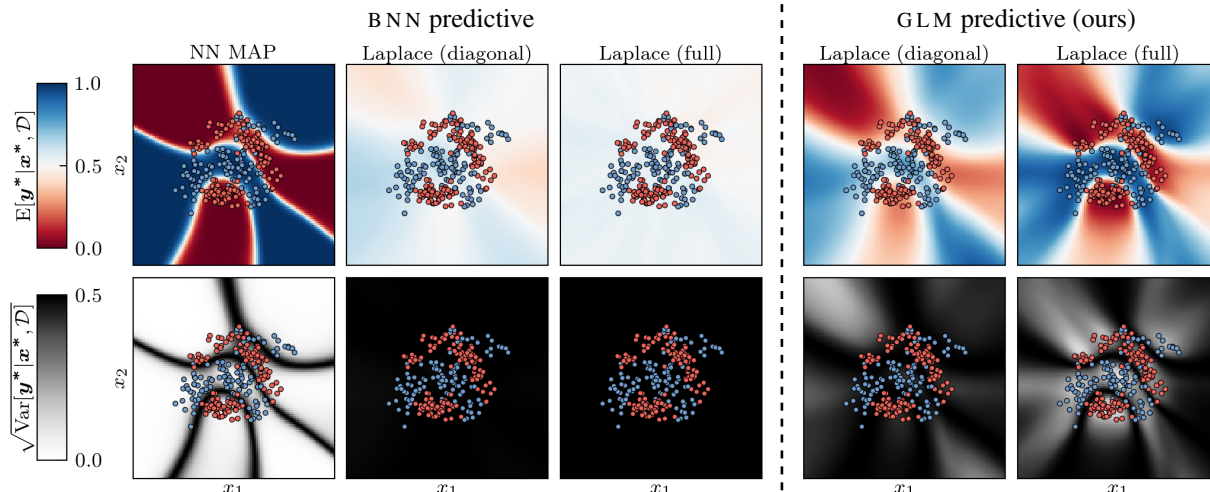


Figure 2. Mean and std dev of the predictive distribution on the banana dataset. Our proposed GLM predictive utilizes the GGN-linearized likelihood and the Laplace posterior (either diagonal or full covariance) for predictions and shows improved growing uncertainties away from the data compared to the MAP. It also alleviates the underfitting problem (Ritter et al., 2018) of the BNN predictive with Laplace approximation that uses the same posterior but the (mismatched) original BNN likelihood. Further results in App. B.2.

proximate inference, see e.g. Rasmussen & Williams (2006, Chapter 3.5). While both formulations are equivalent, they offer two different ways of reducing computational complexity. In particular, for a full posterior covariance approximation, the $\mathcal{O}(P^3)$ operations on the Jacobians are replaced by a $\mathcal{O}(N^3)$ kernel matrix inversion, which enables us to scale to large network architectures. DNN2GP (Khan et al., 2019) uses this same construction to relate approximate inference in DNNs to GPs with neural tangent kernels (Jacot et al., 2018) but always results in a Gaussian likelihood.

2.6. Generalizing previous results for regression

For Gaussian likelihoods, we recover the “linearized Laplace” model by Foong et al. (2019) as well as DNN2GP (Khan et al., 2019) when assuming that neural network training has converged and $\theta^* = \theta_{\text{MAP}}$. In this case, the GLM predictive can be computed analytically instead of by sampling. We highlight that applying the GGN first was key for this derivation – there is nothing special about the subsequent Laplace approximation and any posterior approximation in the GLM can be used instead. These results explain why the GLM predictive outperforms the BNN predictive in Foong et al. (2019) and how underfitting in Ritter et al. (2018) can be addressed.

3. Experiments

The GLM predictive for $\theta^* = \theta_{\text{MAP}}$ on regression is equivalent to the results in Foong et al. (2019) and Khan et al. (2019). Therefore, we focus on classification here.

First, we optimize the MAP objective Eq. (1) and obtain $\theta^* \approx \theta_{\text{MAP}}$. We then linearize the network around θ^*

and employ either a Laplace or Gaussian variational approximation to the posterior of this GLM. As discussed in Section 2.3 we found that the mode μ of the resulting approximate posterior $q(\theta) = \mathcal{N}(\mu, \Sigma)$ could be somewhat different from θ^* (in particular for larger networks). The BNN predictive uses a Laplace approximation (with GGN-approximated Hessian) around θ^* as posterior (Foresee & Hagan, 1997; Ritter et al., 2018). We stress that these Laplace posteriors are very similar and this choice does not influence the qualitative results.

In all experiments, we use a diagonal prior, $p(\theta) = \mathcal{N}(0, \delta^{-1}I_P)$, with precision δ chosen through a validation set. We compare the GLM/GGP predictive that utilises the GGN-linearized likelihood, Eq. (8) (“GLM Laplace/VI”), to the commonly used BNN predictive, Eq. (3) (“BNN Laplace”), as well as to just using θ^* as point estimate (“NN MAP”) and to MFVI maximizing the BNN ELBO with *Bayes by backprop* (“BBB”) (Blundell et al., 2015).

3.1. Illustrative example

We illustrate our method on the banana binary classification dataset with 2d input space, see Fig. 2. We use a 2-layer network with 50 \tanh units (further experiments, comparisons and details in App. B.2).

The BNN predictive with Laplace posterior severely underfits like previously reported (Ritter et al., 2018); underfitting is worse for the full covariance posterior compared to the diagonal. Using the same Laplace posterior but the GLM predictive instead alleviates this problem. Thus, the underfitting is not due to the Laplace posterior but to using a mismatched predictive model. In contrast to using the MAP point-estimate, our GLM predictive also leads to growing

Dataset	NN MAP	BBB	BNN Lapl	GLM Lapl	GLM Lapl diag	GLM VI	GLM VI diag
australian	0.32 \pm 0.01	0.32 \pm 0.01	0.38 \pm 0.00	0.32 \pm 0.01	0.33 \pm 0.01	0.32 \pm 0.01	0.31 \pm 0.01
cancer	0.15 \pm 0.03	0.11 \pm 0.01	0.15 \pm 0.00	0.10 \pm 0.01	0.11 \pm 0.01	0.17 \pm 0.05	0.16 \pm 0.05
ionosphere	0.32 \pm 0.02	0.27 \pm 0.02	0.45 \pm 0.00	0.29 \pm 0.01	0.35 \pm 0.01	0.27 \pm 0.02	0.26 \pm 0.02
glass	0.91 \pm 0.03	0.90 \pm 0.06	1.25 \pm 0.00	0.87 \pm 0.01	0.96 \pm 0.01	0.75 \pm 0.02	0.79 \pm 0.03
vehicle	0.433 \pm 0.007	0.394 \pm 0.005	0.795 \pm 0.003	0.451 \pm 0.004	0.587 \pm 0.004	0.396 \pm 0.007	0.421 \pm 0.006
waveform	0.338 \pm 0.004	0.360 \pm 0.007	0.434 \pm 0.002	0.342 \pm 0.004	0.368 \pm 0.003	0.339 \pm 0.004	0.343 \pm 0.005
digits	0.086 \pm 0.003	0.137 \pm 0.004	0.671 \pm 0.002	0.256 \pm 0.002	0.401 \pm 0.003	0.170 \pm 0.011	0.143 \pm 0.004
satellite	0.222 \pm 0.002	0.274 \pm 0.002	0.429 \pm 0.001	0.240 \pm 0.001	0.281 \pm 0.001	0.219 \pm 0.001	0.238 \pm 0.002

Table 1. Negative test log likelihood (lower is better) on UCI binary/multiclass classification tasks with 1 hidden layer (50 tanh). Approximate inference in the GLM is strictly superior to a Laplace approximation in the BNN and also provides a good alternative to Bayes-by-Backprop (Blundell et al., 2015). Results on accuracy and calibration can be found in Appendix B.3.

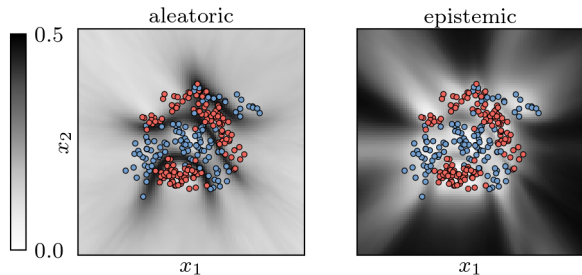


Figure 3. Aleatoric and epistemic uncertainty for GLM Laplace.

predictive variances away from the data in line with previous observations in the regression case (Foong et al., 2019; Khan et al., 2019). Moreover, the predictive variance decomposes into meaningful aleatoric and epistemic uncertainties (Kwon et al., 2020), see Fig. 3.

3.2. UCI classification

We now compare the different methods on a set of UCI classification tasks on a network with 1 and 2 hidden layer of 50 tanh and relu units. In Tab. 1, we report the test log predictive probabilities over 10 splits (70% train/15% valid/15% test) for 1 hidden tanh layer (further results and experimental details in App. B.3). The optimal prior precision δ is tuned on a validation set for each split and method.

The results clearly show again that the predictive distribution has to match inference: when the GGN approximation is used, the GLM predictive (“GLM Lapl”) clearly outperforms the BNN predictive (“BNN Lapl”). Applying variational inference to the GLM (“GLM VI”) can further boost performance, and our methods perform on par or better than MFVI. Further, our method works out of the box for multiple hidden layers where MFVI is hard to tune.

In Fig. 4 we highlight that the GLM predictive consistently outperforms the BNN predictive for any setting of the prior precision hyperparameter and that GLM VI can improve over the MAP estimate.

3.3. Larger scale results on CIFAR10

Finally, we evaluate our method on a larger problem, CIFAR10 with LeNet5 network. Due to the size of the model,

we perform the inference in function space with the GGP instead of GLM. We split the original training set into 10 splits of 5000 examples each and evaluate performance of GGP VI and Laplace, see Tab. 2. For Laplace, we compare the GGP with mode at θ^* , i.e. \mathbf{f}^* , and with mode at an improved estimate $\mathbf{f}_{\text{GGP mode}}$ obtained by several update steps through approximate inference in the linearized model.

While GGP Laplace evaluated with mode at \mathbf{f}^* shows comparable performance to the NN MAP, several approximate update steps to $\mathbf{f}_{\text{GGP mode}}$ result in a GGP model with considerably improved performance. This might be because it is hard to fully converge to θ_{MAP} when training a large, highly non-linear NN model, but it is easier to find the mode of a convex posterior of a GLM. GGP VI with inducing points attains equivalent predictive performance to GGP Laplace.

	test nll	test accuracy
NN MAP (at θ^*)	1.415 \pm 0.003	49.7% \pm 0.2%
GGP Laplace (\mathbf{f}^*)	1.407 \pm 0.003	49.6% \pm 0.1%
GGP Laplace ($\mathbf{f}_{\text{GGP mode}}$)	1.366 \pm 0.006	52.0% \pm 0.2%
GGP VI, 512 ind. points	1.409 \pm 0.007	50.6% \pm 0.1%
GGP VI, 1024 ind. points	1.381 \pm 0.007	51.4% \pm 0.2%
GGP VI, 2048 ind. points	1.368 \pm 0.007	51.9% \pm 0.2%

Table 2. CIFAR10 with LeNet5

4. Conclusion

In this paper we argued that in Bayesian deep learning, the frequently utilized generalized Gauss-Newton (GGN) approximation should be understood as a modification of the underlying probabilistic model and should be considered separately from further approximate inference techniques. Applying the GGN approximation turns a Bayesian neural network (BNN) into a locally linearized generalized linear model or, equivalently, a generalized Gaussian process. Because we then use this linearized model for inference, we should also predict using this modified likelihood rather than the original BNN likelihood. This formulation extends previous results by Khan et al. (2019) and Foong et al. (2019) to general likelihoods and alleviates underfitting behaviour observed e.g. by Ritter et al. (2018). We demonstrate our approach on UCI classification datasets as well as CIFAR10. In future work we aim to scale our approach further.

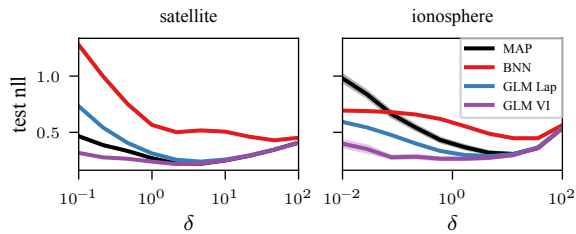


Figure 4. The GLM predictive (—/—) outperforms the BNN predictive (—) for all settings of the model hyperparameter δ .

Acknowledgements

We thank Emtiyaz Khan for the many fruitful discussions that lead to this work as well as Michalis Titsias for feedback on the manuscript. We are also thankful for the RAIDEN computing system and its support team at the RIKEN AIP.

References

- Amari, S.-I. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.
- Bishop, C. M. *Pattern recognition and machine learning*. springer, 2006.
- Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pp. 1613–1622, 2015.
- Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *Siam Review*, 60(2): 223–311, 2018.
- Foong, A. Y., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. ‘in-between’ uncertainty in bayesian neural networks. *arXiv preprint arXiv:1906.11537*, 2019.
- Foresee, F. D. and Hagan, M. T. Gauss-newton approximation to bayesian learning. In *Proceedings of International Conference on Neural Networks (ICNN’97)*, volume 3, pp. 1930–1935. IEEE, 1997.
- Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with GPU acceleration. *CoRR*, abs/1809.11165, 2018.
- Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D., and Eslami, S. M. A. Conditional neural processes. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1704–1713. PMLR, 2018.
- Hensman, J., Rattray, M., and Lawrence, N. D. Fast variational inference in the conjugate exponential family. In *Advances in neural information processing systems*, pp. 2888–2896, 2012.
- Hensman, J., Matthews, A., and Ghahramani, Z. Scalable variational gaussian process classification. 2015.
- Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pp. 8571–8580, 2018.
- Khan, M., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. Fast and scalable bayesian deep learning by weight-perturbation in adam. In *International Conference on Machine Learning*, pp. 2611–2620, 2018.
- Khan, M. E. E., Immer, A., Abedi, E., and Korzepa, M. Approximate inference turns deep networks into gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 3088–3098, 2019.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems*, pp. 2575–2583, 2015.
- Kwon, Y., Won, J.-H., Kim, B. J., and Paik, M. C. Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation. *Computational Statistics & Data Analysis*, 142: 106816, 2020. ISSN 0167-9473.
- Lawrence, N. D. *Variational Inference in Probabilistic Models*. University of Cambridge, 2001.
- Martens, J. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.
- Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417, 2015.
- Naeini, M. P., Cooper, G., and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
- Rasmussen, C. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- Rasmussen, C. E. and Nickisch, H. Gaussian processes for machine learning (gpml) toolbox. *Journal of Machine Learning Research*, 11:3011–3015, 2010.

Ritter, H., Botev, A., and Barber, D. A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations*, 2018.

Titsias, M. Variational learning of inducing variables in sparse gaussian processes. In van Dyk, D. and Welling, M. (eds.), *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 567–574. PMLR, 16–18 Apr 2009.

Williams, C. K. I. and Barber, D. Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pp. 5852–5861, 2018.

A. Derivations

A.1. Impact of Gauss-Newton on the probabilistic model

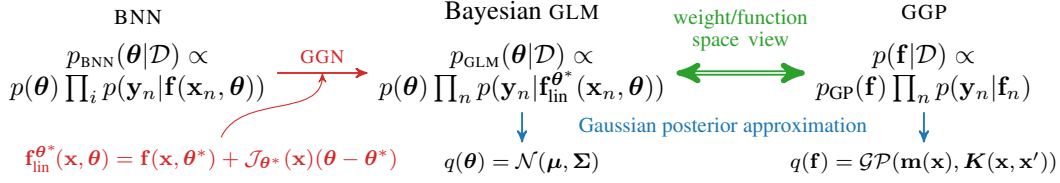


Figure A1. The generalized Gauss Newton approximation (GGN) turns a Bayesian neural network (BNN) into a generalized linear model (GLM) with same prior and likelihood distribution, but network function $\mathbf{f}(\mathbf{x}_i, \boldsymbol{\theta})$ linearized around $\boldsymbol{\theta}^*$ (\rightarrow). The GLM is equivalent to a generalized Gaussian process (GGP) (\leftrightarrow). Inference is made tractable with a Gaussian posterior approximation (\rightarrow).

The combination of a Laplace (Foresee & Hagan, 1997; Ritter et al., 2018) or a Gaussian variational (Khan et al., 2018; Zhang et al., 2018) approximation with the Generalized Gauss-Newton (GGN) approximation gives rise to recently successful approximate inference algorithms. In the aforementioned algorithms, the GGN facilitates an approximation to the Hessian of the log likelihood with respect to the neural network parameters $\boldsymbol{\theta}$. The Hessian can be decomposed over the N data points due to independence, such that we focus on a representative input-output pair $\mathbf{x} \in \mathbb{R}^D, \mathbf{y} \in \mathbb{R}^C$.

Recall from Section 2.1, that we define the *Jacobian* $\mathcal{J} \in \mathbb{R}^{C \times P}$ and the *Hessian* $\mathcal{H} \in \mathbb{R}^{C \times P \times P}$ of the neural network \mathbf{f} w.r.t. its parameters $\boldsymbol{\theta}$ as

$$[\mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x})]_{ci} = \frac{\partial f_c(\mathbf{x}, \boldsymbol{\theta})}{\partial \theta_i} \quad (\text{A1})$$

$$[\mathcal{H}_{\boldsymbol{\theta}}(\mathbf{x})]_{cij} = \frac{\partial^2 f_c(\mathbf{x}, \boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} \quad (\text{A2})$$

Further, we define the gradient of the log likelihood w.r.t. the neural network features \mathbf{f} as $[\mathbf{r}(\mathbf{y}; \mathbf{f})]_c = \frac{\partial \log p(\mathbf{y}|\mathbf{f})}{\partial f_c}$ and the Hessian of the log likelihood w.r.t. the neural network features \mathbf{f} as $[\boldsymbol{\Lambda}(\mathbf{y}; \mathbf{f})]_{ck} = -\frac{\partial^2 \log p(\mathbf{y}|\mathbf{f})}{\partial f_c \partial f_k}$, respectively. Then, we can express the Jacobian and Hessian of the log likelihood w.r.t. the neural network parameters $\boldsymbol{\theta}$ by using the chain rule as:

$$\nabla_{\boldsymbol{\theta}} \log p(\mathbf{y}|\mathbf{f}(\mathbf{x}, \boldsymbol{\theta})) = \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x})^T \mathbf{r}(\mathbf{y}; \mathbf{f}) \quad (4)$$

$$\nabla_{\boldsymbol{\theta}\boldsymbol{\theta}}^2 \log p(\mathbf{y}|\mathbf{f}(\mathbf{x}, \boldsymbol{\theta})) = \mathcal{H}_{\boldsymbol{\theta}}(\mathbf{x})^T \mathbf{r}(\mathbf{y}; \mathbf{f}) - \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x})^T \boldsymbol{\Lambda}(\mathbf{y}; \mathbf{f}) \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x}), \quad (5)$$

The Gauss-Newton approximation to the Hessian of the log likelihood w.r.t. neural network parameter $\boldsymbol{\theta}$ can then be written as (Martens, 2014)

$$\nabla_{\boldsymbol{\theta}\boldsymbol{\theta}}^2 \log p(\mathbf{y}|\mathbf{f}(\mathbf{x}, \boldsymbol{\theta})) \approx -\mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x})^T \boldsymbol{\Lambda}(\mathbf{y}; \mathbf{f}) \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x}). \quad (\text{A3})$$

This approximation is exact if all residuals are zero, i.e., $\mathbf{r}(\mathbf{y}_n; \mathbf{f}(\mathbf{x}_n; \boldsymbol{\theta})) = 0 \forall n \in \{1, \dots, N\}$. This typically does not hold in practice and is also not desirable as it indicates overfitting.

Alternatively and equivalently, one can interpret the Gauss-Newton approximation as the true Hessian of a log likelihood of a locally linearized neural network (Martens & Grosse, 2015; Bottou et al., 2018). To show this, let $\boldsymbol{\theta}^*$ be the parameter at which we want to evaluate the above approximation to the Hessian. The linearized neural network at $\boldsymbol{\theta}^*$ is given by

$$\mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{f}(\mathbf{x}, \boldsymbol{\theta}^*) + \mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x})(\boldsymbol{\theta} - \boldsymbol{\theta}^*), \quad (\text{A4})$$

and we obtain for its full (un-approximated) Hessian of the log likelihood

$$\nabla_{\boldsymbol{\theta}\boldsymbol{\theta}}^2 \log p(\mathbf{y}|\mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}, \boldsymbol{\theta})) = -\mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x})^T \boldsymbol{\Lambda}(\mathbf{y}; \mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}, \boldsymbol{\theta})) \mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x}), \quad (\text{A5})$$

where the first term in Eq. (5) vanishes because the network is linear. Thus, we find that the GGN-Hessian approximation in Eq. (A3) and the un-approximated Hessian of the locally linearized model at $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ in Eq. (A5) are equivalent; the Jacobians become $\mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x})$ in both cases and $\mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}; \boldsymbol{\theta}^*) = \mathbf{f}(\mathbf{x}; \boldsymbol{\theta}^*)$.

This second view allows us to clearly understand the impact of the GGN on the underlying probabilistic model. The linearization reduces the BNN to a Bayesian generalized linear in the parameters model (GLM), see \rightarrow in Fig. A1.

We now apply the linearization that leads to the GGN approximation of the Hessian in our probabilistic neural network model. The joint of the original BNN model is given by

$$p_{\text{BNN}}(\boldsymbol{\theta}, \mathcal{D}) = p(\boldsymbol{\theta}) \prod_{n=1}^N p(\mathbf{y}_n | \mathbf{f}(\mathbf{x}_n; \boldsymbol{\theta})). \quad (\text{A6})$$

Applying the local linearization at some parameter $\boldsymbol{\theta}^*$, we obtain the following modified probabilistic model:

$$p_{\text{GLM}}(\boldsymbol{\theta}, \mathcal{D}) = p(\boldsymbol{\theta}) \prod_{n=1}^N p(\mathbf{y}_n | \mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}_n, \boldsymbol{\theta})). \quad (\text{A7})$$

A.2. Recovering the Laplace-GGN

Suppose $\boldsymbol{\theta}^* \approx \arg \max_{\boldsymbol{\theta}} \log p_{\text{BNN}}(\boldsymbol{\theta}, \mathcal{D})$, i.e., $\boldsymbol{\theta}^*$ is a MAP estimate of the BNN model, Eq. (A6). To construct a Laplace approximation, Ritter et al. (2018) and Foresee & Hagan (1997) approximate the Hessian of the log joint distribution at $\boldsymbol{\theta}^*$. In their derivation, both make use of the GGN as either a Hessian approximation or optimization algorithm and therefore do not identify the underlying formulation of the GLM. They approximate the Hessian of the BNN log joint as follows:

$$\nabla_{\boldsymbol{\theta}\boldsymbol{\theta}}^2 \log p_{\text{BNN}}(\boldsymbol{\theta}, \mathcal{D})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} \approx \nabla_{\boldsymbol{\theta}\boldsymbol{\theta}}^2 \log p(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} + \sum_{n=1}^N \mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x})^\top \Lambda(\mathbf{y}_n; \mathbf{f}(\mathbf{x}_n; \boldsymbol{\theta}^*)) \mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x}) \quad (\text{A8})$$

$$= \nabla_{\boldsymbol{\theta}\boldsymbol{\theta}}^2 \log p_{\text{GLM}}(\boldsymbol{\theta}, \mathcal{D})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}, \quad (\text{A9})$$

where the last equality emphasizes that the Hessian approximation used is in fact equivalent to that of our GLM formulation. Therefore, when we care about the underlying probabilistic model we should make use of the GLM as we have a proper Laplace approximation of it. In contrast, we do not know how good of an approximation we obtain in Eq. (A8) unless we know that all residuals are zero. In classification problems with GLM likelihoods, the residuals can only become numerically zero due to the natural sigmoid inverse link function mapping to $(0, 1)$ and the labels being in $0, 1$. Critically, while in optimization we are interested in how the GGN Hessian impacts convergence in second-order algorithms, in approximate inference we need to estimate the impact on the underlying model.

A.3. From GLM to Generalized Gaussian process model (\leftrightarrow in Fig. A1)

The identification of an underlying GLM in approximate inference further allows specification of an equivalent Gaussian process (GPP) model. The parametric prior distribution $p(\boldsymbol{\theta})$ induces a functional distribution on $\mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}, \boldsymbol{\theta})$:

$$\mathbf{f} \sim \mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}, \boldsymbol{\theta}_m) = \mathbf{f}(\mathbf{x}, \boldsymbol{\theta}^*) + \mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x})(\boldsymbol{\theta}_m - \boldsymbol{\theta}^*) \quad \text{with} \quad \boldsymbol{\theta}_m \sim p(\boldsymbol{\theta}), \quad (\text{A10})$$

which induces a Gaussian process in \mathbf{f} because we assume a Gaussian prior $p(\boldsymbol{\theta})$. Computing the mean and covariance under the Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{m}_0, \mathbf{S}_0)$ allows us to obtain the mean and covariance function of the Gaussian process prior

$$\begin{aligned} \mathbf{m}(\mathbf{x}) &= \mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}; \mathbf{m}_0) \\ \mathbf{k}(\mathbf{x}, \mathbf{x}') &= \mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x}) \mathbf{S}_0 \mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x}')^\top, \end{aligned} \quad (9)$$

which define the GP prior $p(\mathbf{f}) = \mathcal{GP}(\mathbf{m}(\mathbf{x}), \mathbf{k}(\mathbf{x}, \mathbf{x}'))$. This Gaussian process prior is equivalent to the Gaussian process prior of DNN2GP and similar to the neural tangent kernel (Khan et al., 2019; Jacot et al., 2018). The key difference is that we define a GPP model as opposed to a Gaussian process regression model and can therefore handle non-Gaussian likelihoods in contrast to previous approaches. The GPP model is then given by the joint distribution $p(\mathbf{f}, \mathcal{D}) = p(\mathbf{f}) \prod_{n=1}^N p(\mathbf{y}_n | \mathbf{f}_n)$. As for the GLM, we can approximately infer this model using a Laplace approximation (Williams & Barber, 1998; Rasmussen & Williams, 2006) or variational approximation (Titsias, 2009; Hensman et al., 2012; 2015).

A.4. GLM vs. GPP

A naive Gaussian posterior approximation to either the GLM or the GPP typically has computational costs of $\mathcal{O}(P^3)$ or $\mathcal{O}(N^3)$, respectively. In practice, however, we can employ diagonal or structured posterior covariance approximations to

reduce the cost from $\mathcal{O}(P^3)$ down to $\mathcal{O}(P)$ when working with the GLM. As the experiments suggest, using a diagonal variational approximation for the GLM is computationally cheaper and also performs decently well and is well-calibrated. Nonetheless, our experiments suggest that a full covariance leads to improvements over a diagonal covariance. In particular, the full covariance posterior approximation is well-calibrated compared to the diagonal version.

In contrast, scaling approximate inference in a GGP typically goes down other routes: For the Laplace approximation, we can reduce the computational cost to $\mathcal{O}(NM^2)$ using, for example, the Nyström approximation. Similarly, a variational approximation (Titsias, 2009) can make use of $M \ll N$ inducing points to make inference scalable as $\mathcal{O}(NM^2)$ and is typically superior to the Nyström approximation. Recent advances in scalable GP inference allow us to evaluate approximations on minibatches and scale as $\mathcal{O}(\tilde{N}M^2)$, where \tilde{N} denotes the minibatch size (Hensman et al., 2012).

Our GGP formulation enables us to do functional inference for parametric Bayesian neural networks with general likelihoods and therefore presents an alternative to other recent approaches like “neural processes” (Garnelo et al., 2018). Interestingly, the parametric GLM and functional GGP both offer different posterior approximations that enable scalability. This can open up new ways for highly flexible functional posterior approximations combining neural networks and Gaussian process inference.

B. Experimental details and additional results

B.1. Inference in GLM and GGP

GLM Laplace: We use the Adam optimizer with learning rate 10^{-3} and run for 1000 iterations using full-batch gradient descent in all experiments.

GLM VI: We use natural gradient variational inference (Amari, 1998; Khan et al., 2018; Zhang et al., 2018) to optimize the mean and covariance of the variational Gaussian posterior approximation. We take 250 full batch iterations in all experiments and use the learning rate 10^{-2} for diagonal covariance and 10^{-3} for full covariance.

GGP Laplace: Following (Williams & Barber, 1998; Rasmussen & Williams, 2006), we optimize for $\mathbf{f}^{\theta_{\text{MAP}}}$ using Newton’s method, combining it with a line search similar to how it is implemented in the `gpm1` toolbox (Rasmussen & Nickisch, 2010). For multiclass classification, we assume independence of prior functions across the outputs to reduce the computational complexity per step from $\mathcal{O}(C^3N^3)$ to $\mathcal{O}(CN^3)$ (Rasmussen & Williams, 2006).

GGP VI: We use VI inference with “whitening” and inducing points available in `GPYTORCH` (Gardner et al., 2018). It assumes independence of both prior and posterior functions across the outputs to reduce the computational complexity per update to $\mathcal{O}(C\tilde{N}M^2)$. We optimize the ELBO using Adam optimizer with initial learning rate 10^{-1} . We do not learn the inducing points, fixing them to a random subset of the training data.

B.2. Illustrative example

We use a synthetic dataset known as ‘banana’ and separate 5% ($N = 265$) of it as training as 5% as validation dataset. For NN MAP, we tune the prior precision δ using the validation dataset on a uniformly-spaced grid of 10 values in range $[0.1, 2.0]$ for all architectures with at least two layers, otherwise we use a smaller range $[0.02, 0.4]$; for BBB (Blundell et al., 2015) we use 10 log-spaced values between 10^{-3} and 1. We optimize the models using full-batch gradient descent with the Adam optimizer with initial learning rate 10^{-2} (NN MAP) and 10^{-1} (BBB) for 3000 epochs, decaying the learning rate by a factor of 10 after 2400 and 2800 epochs. We run inference in GGP for 1000 iterations without inducing points as the training dataset is very small. After 500 epochs, we start decaying the learning rate by a factor of 0.99 each epoch. We show the predictive distribution for a 100×100 grid with input features x_1 and x_2 in $[-4, 4]$ range using 1000 Monte Carlo-samples to estimate the posterior predictive distribution for all methods.

In machine learning, uncertainty is often classified into aleatoric and epistemic. The aleatoric uncertainty is due to inherent noise in the data (e.g. two or more different classes overlapping in the input domain) and will always be there regardless of how many data points we sample. Therefore we might be able to quantify this uncertainty better by sampling more data, but we will not be able to reduce it. On the other hand, epistemic uncertainty is caused by lack of knowledge and can be minimized by sampling more data.

Therefore, the decomposition into aleatoric and epistemic uncertainty allows to establish to what extent a model is uncertain about its predictions due to inherent noise in the data and to what extent the uncertainty is due to the lack of data. Such

a distinction can be helpful in certain areas of machine learning such as active learning. To decompose the uncertainty of a Bernoulli variable, we follow (Kwon et al., 2020) which derives the following split of total variance into aleatoric and epistemic uncertainty:

$$\text{Var}_{p_{\text{GLM}}(\mathbf{y}^*|\mathbf{x}^*,\mathcal{D})}(\mathbf{y}^*) = \underbrace{\int \text{Var}_{p_{\text{GLM}}(\mathbf{y}^*|\mathbf{x}^*,\boldsymbol{\theta})}(\mathbf{y}^*)q(\boldsymbol{\theta}) d\boldsymbol{\theta}}_{\text{aleatoric}} + \underbrace{\int [\mathbb{E}_{p_{\text{GLM}}(\mathbf{y}^*|\mathbf{x}^*,\boldsymbol{\theta})}(\mathbf{y}^*) - \mathbb{E}_{p_{\text{GLM}}(\mathbf{y}^*|\mathbf{x}^*,\mathcal{D})}(\mathbf{y}^*)]^2 q(\boldsymbol{\theta}) d\boldsymbol{\theta}}_{\text{epistemic}} \quad (\text{B11})$$

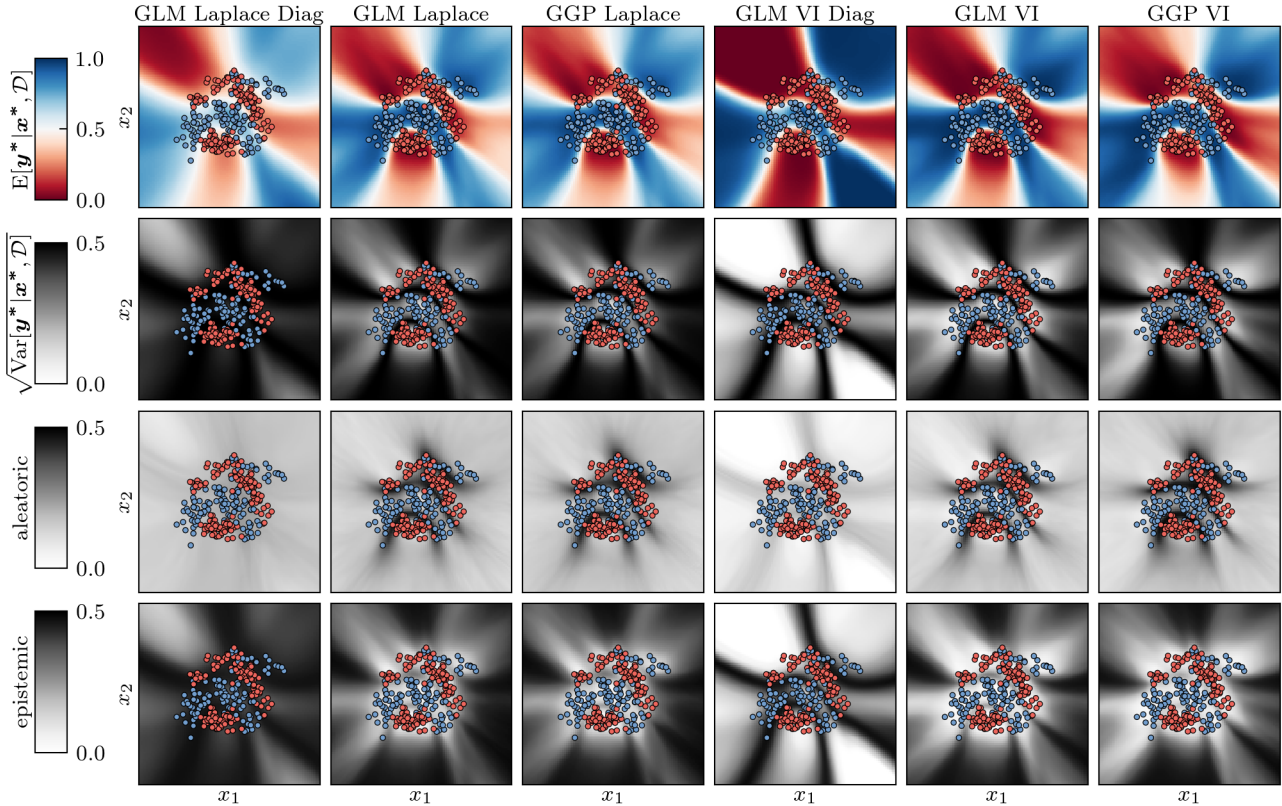


Figure B2. Mean, std dev, aleatoric and epistemic uncertainty of the predictives on the banana dataset for all proposed methods for a network with **2 hidden layers with 50 units and tanh activation function**. Laplace posterior approximation provides less confident predictions than VI-based one. Diagonal approximations (GLM Laplace and GLM VI) provide much worse uncertainty estimates than their full covariance counterparts. For diagonal Laplace approximation, the predictions get underconfident and miscalibrated, i.e. the uncertainty decreases away from data in many directions. On the other hand, for diagonal VI approximation, the predictions away from data get more confident when comparing to VI approximation with full covariance. Both diagonal approximations fail to capture aleatoric uncertainty. As expected, GGP predictives look very similar to GLM predictives.

In Fig. B2 we compare all proposed methods: GLM/GGP with Laplace/VI approximations. For GLM, we additionally show diagonal approximations. Because function space and weight space view are equivalent up to approximation, the GGP predictives look very similar to the corresponding GLM predictives. The Laplace approximation results in much less confident predictions than the VI approximation. This is expected as the Laplace approximation fits a Gaussian distribution locally at a mode of the true posterior, which can be highly skewed for classification likelihoods. As a result, when sampling from the Laplace approximation there are many samples with low posterior probability which leads to overestimated variance. On the other hand, VI fits a Gaussian approximation globally, avoiding sampling very improbable parameters / functions. Furthermore, the comparison with diagonal approximations shows that full covariance might be necessary to provide reasonable uncertainty estimates. The diagonal approximation fails to capture aleatoric uncertainty and results in greatly overestimated variance for Laplace approximation, even in the region of the input space densely populated by data. On the other hand, for VI, it seems to underestimate the uncertainty.

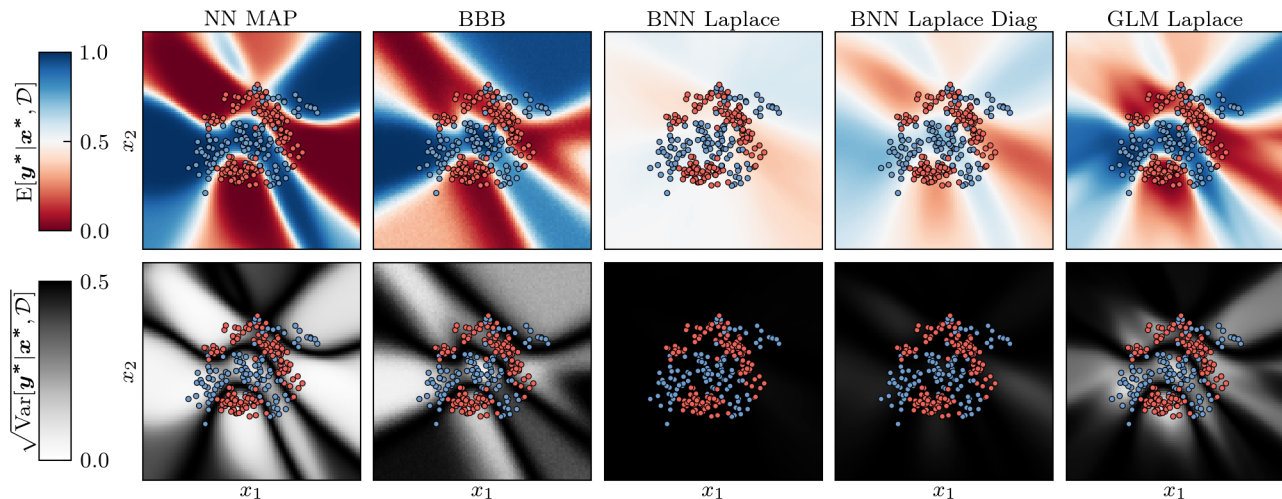


Figure B3. Mean and std dev of the predictions on the banana dataset for a network with **1 hidden layer with 50 units and tanh activation function**. A significant decrease of parameters compared to a network with 2 layers results in a decrease in confidence of NN MAP as well as the corresponding GLM predictions. BNN Laplace predictive improves slightly compared to predictions in a model with 2 hidden layers, but the variance is still greatly overestimated.

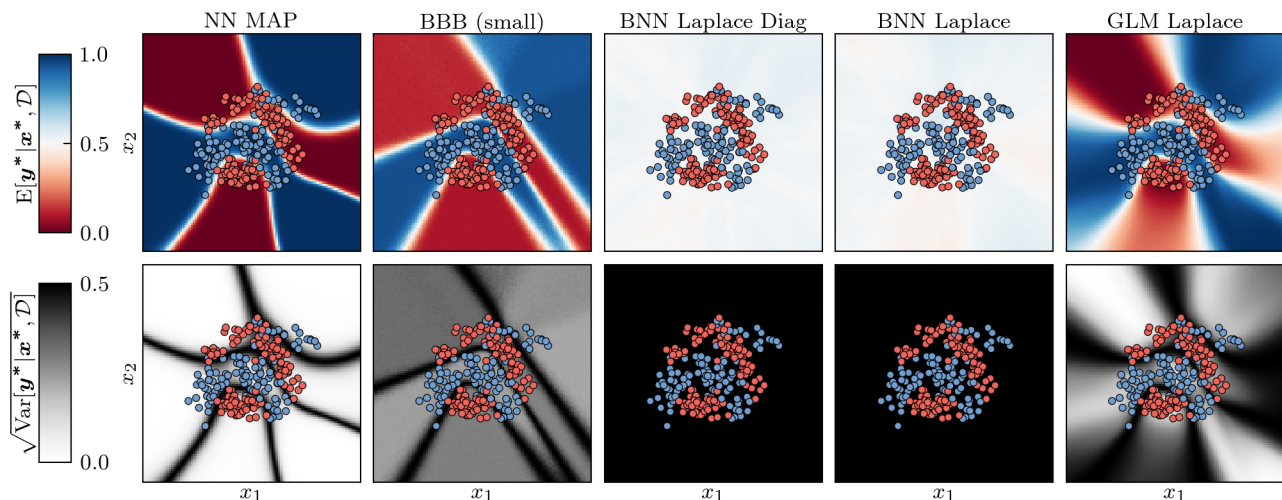


Figure B4. Mean and std dev of the predictions on the banana dataset for a network with **3 hidden layers with 50 units each and tanh activation function** (with the exception of BBB where number of hidden units per layer was limited to 5 due to failing to fit the data with more units per layer). Increasing the model capacity leads to more confident predictions in NN MAP and the corresponding GLM models (narrower decision boundary), yet the GLM model is still able to produce reasonable uncertainties growing away from data.

In Figs. B3 and B4, we show the NN MAP, BBB, BNN Laplace, and GLM Laplace predictives for a network with one and three layers, respectively. A lower number of parameters slightly improves the performance of BNN Laplace, but still the variance is severely overestimated. The GLM Laplace predictive has higher variance compared to the model with two layers (Fig. 2) similarly to NN MAP on which it was based. When the model is greatly overparameterized at 3 layers, BNN Laplace completely fails even for the diagonal approximation. For GLM Laplace, similarly as for NN MAP, the confidence of the predictions increases, however, the former still produces reasonable uncertainties growing away from data.

In Fig. B5, we show the predictives for `relu` activation function. GLM predictives show that both mean and variance are sliced by linear surfaces as can be expected from a piece-wise linear activation function. However, even though the predictive variance still decomposes reasonably into aleatoric and epistemic uncertainty, the overall quality and consistency of the predictive distribution in the context of this classification problem seems much lower than for `tanh` activation function. This suggests that the activation function may play an important role in ensuring the quality of predictive distributions for BNNs.

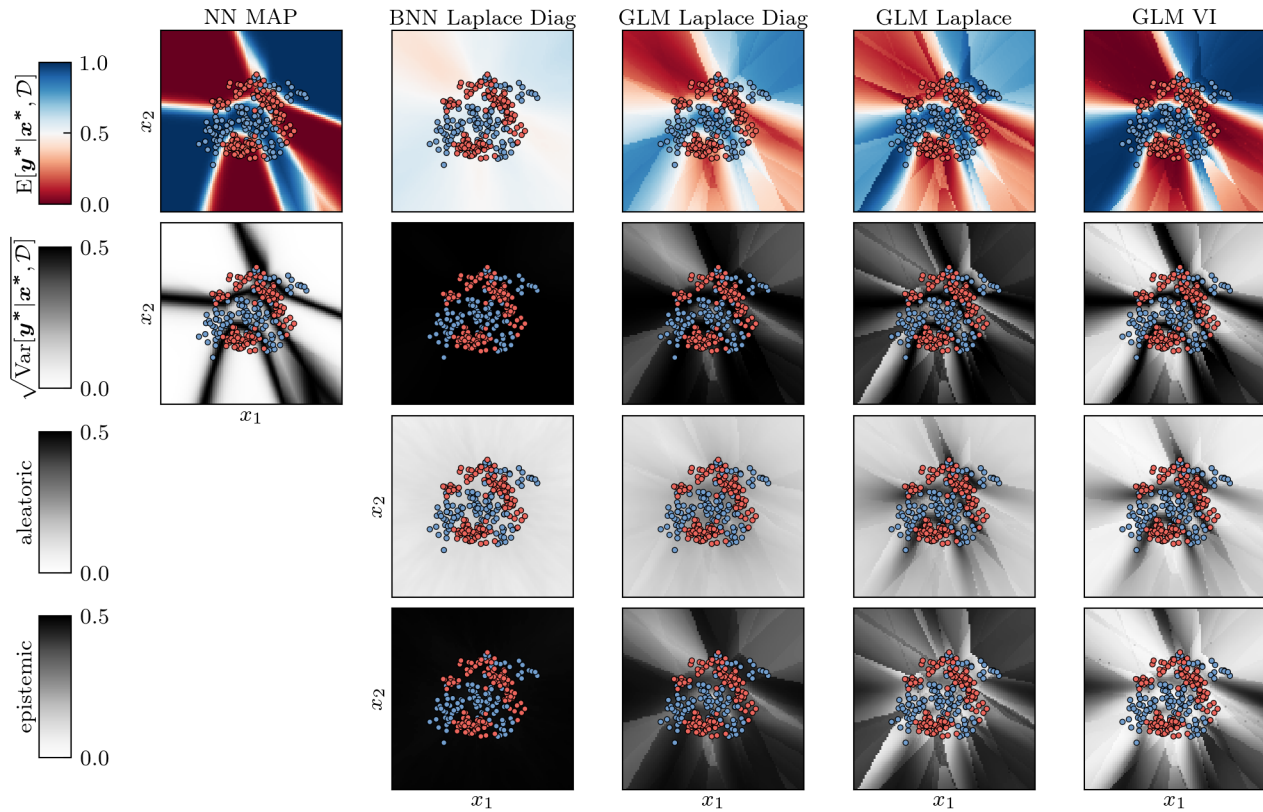


Figure B5. Mean and std dev of the predictions on the banana dataset for a network with **2 hidden layers with 50 units and relu activation function**.

Finally, in Fig. B6 we show our GLM Laplace and VI predictions cope with quantifying the uncertainty in the regions of space without any data, but surrounded by data - so-called in-between uncertainty as shown for regression in (Foong et al., 2019). In order to produce a 'data gap', we remove all the data points with $x_1 \in [-0.5, 0.5]$ from training and validation datasets. Our results suggest that full covariance is necessary to capture this kind of uncertainty. Both GLM Laplace and VI predictions show a considerable increase in epistemic uncertainty in $x_1 \in [-0.5, 0.5]$ range. NN MAP and mean-field methods (both BBB and GLM with diagonal posterior approximation) fail to capture epistemic uncertainty - the width of the decision boundaries does not change in the in-between data gap.

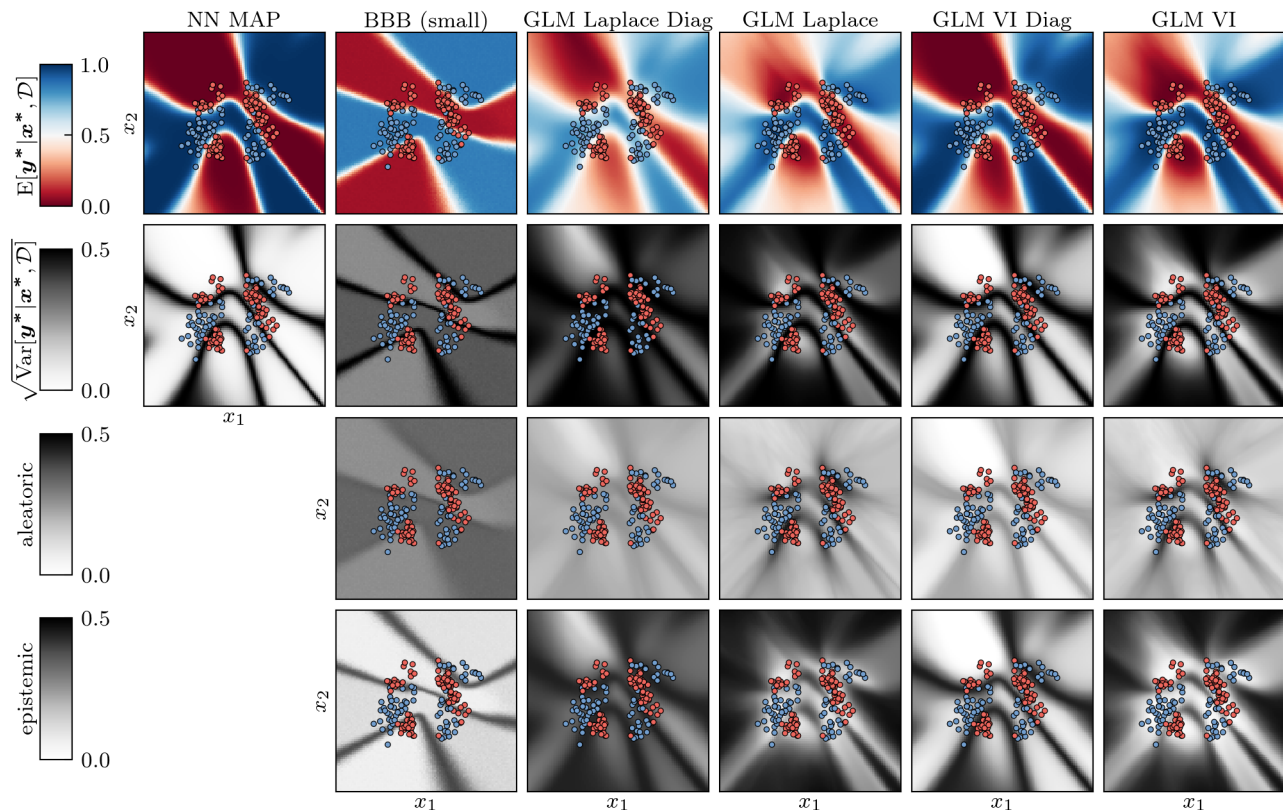


Figure B6. Mean and std dev of the predictives on the banana dataset with data points with $x_1 \in [-0.5, 0.5]$ removed to visualize in-between data uncertainty analogically to regression in (Foong et al., 2019). We used 2 layer network with 50 units and tanh activation function (and only 10 units per layer for BBB). Only full covariance GLM model provides reasonable in-between uncertainties which are clearly visible in epistemic uncertainty - the model is less certain about the decision boundary in $[-0.5, 0.5]$ range due to the lack of data. All mean-field methods (including diagonal GLM model) fail to properly quantify epistemic uncertainty.

B.3. UCI

We test the mentioned methods for approximate inference on 8 UCI classification data sets available from UCI Machine Learning Repository: <https://archive.ics.uci.edu/ml/datasets.php>, see Tab. B1

We compare the neural network MAP, a BNN with mean-field variational inference trained with *Bayes-by-Backprop* (Blundell et al., 2015) (BBB) using the local reparameterization trick (Kingma et al., 2015), and a BNN with Laplace approximation using the GGN (Ritter et al., 2018; Foresee & Hagan, 1997) (Eq. 3) with our methods. We compare both a Laplace and Gaussian variational approximation of the GLM formulation (ref to main section) with both diagonal and full posterior covariance approximations. For all methods, we train until convergence (for BBB 5000 steps, for MAP 10000 steps) with the Adam optimizer (Kingma & Ba, 2014) using a learning rate of 10^{-3} for MAP and 10^{-3} for BBB. For inference in the GLM, we use 1000 iterations for the Laplace approximation and 250 for the variational approximations. For the Laplace approximation, we further optimize the GLM using Adam and a learning rate of 10^{-3} . For the variational approximation the GLM, we use a natural gradient VI algorithm to update the mean and covariance of the posterior approximation (Amari, 1998; Khan et al., 2018; Zhang et al., 2018) with learning rates 10^{-3} for a full and 10^{-2} for a diagonal covariance. We split each dataset 10 times randomly into *train/validation/test* sets with ratios 70%/15%/15% and stratify by the labels to obtain proportional number of samples with particular classes. For each split, we train all above-mentioned methods with 10 different prior precisions δ on a log-spaced grid from 10^{-2} to 10^2 , except for the larger datasets *satellite* and *digits* where the grid is from 10^{-1} to 10^2 . In the resulting tables, we report the performance with the standard error on the test set after selecting the best hyperparameter δ on the validation set. We run above experiment for three network architectures: one hidden layer with *tanh* activation and 50 hidden units and a two-layer network with *tanh* activation and 50 units on each hidden layer. Further, we run the experiment for one hidden layer and *relu* activation function.

Tables B2 and B3 complement Table 1 with results on accuracy and expected calibration error (ECE) which is a metric that

Dataset	number of datapoints N	input dimension D	number of classes C
australian credit approval	690	14	2
breast cancer Wisconsin	569	32	2
ionosphere	351	34	2
glass identification	214	6	10
vehicles silhouettes	846	14	4
waveform	1000	31	3
digits	1797	64	10
satellite	6435	36	6

Table B1. Overview of UCI classification datasets used

tells us the mismatch between classification accuracy and predictive certainty (Naeini et al., 2015). For the ECE, we use 10 bins. Further, Tables B4, B5, and B6 list the corresponding results for two hidden layers with 50 hidden units for each layer and \tanh activation. The two-layer network builds a stark contrast to the single layer network as it has more parameters than data points for all of the benchmarks except on the *satellite* dataset. In Tables B7, B8, and B9 we report the corresponding results for a relu instead of \tanh activation function on the hidden layer. The results are slightly different but overall in favor of the GLM VI method, especially with respect to the calibration performance as shown in Table B9.

In the single layer experiment, the additional results on accuracy and ECE suggest that the MAP generally gives very accurate predictions and is on par with GLM VI both in terms of accuracy and ECE. Despite the suboptimal calibration (according to ECE and test log likelihood), BNN Laplace performs well in terms of accuracy on the *australian* and *breast cancer* datasets. The results for two hidden layers show similar trends but the performance of BBB suffers from higher non-linearity and more parameters. The GLM VI method achieves the best nlls except on the *digits* dataset and is also best calibrated according to the ECE metric. When we swap the \tanh activation in the hidden layer with a relu activation, the results do not change significantly: GLM VI and BBB perform best in the test log likelihood while the MAP and GLM VI perform best in accuracy. Strikingly, GLM VI performs by far best in terms of calibration. Overall, the proposed GLM VI method dominates the benchmarks on the test log likelihood and ECE metrics while the MAP often provides the best accuracies.

The further results suggest that the proposed methods, especially variational inference in the proposed GLM, perform on par or better than MAP and the BBB algorithm for BNNs. In particular for more parameters (see e.g. Tab. B4), the gap between BBB and the GLM grows. While the test negative log likelihood and calibration results for BNN Laplace are worse compared to the other methods, the accuracy can still be on par with other methods in some cases. Generally, the proposed GLM Laplace yields strictly better results than BNN Laplace and alleviates its underfitting problem.

Dataset	NN MAP	BBB	BNN Lapl	GLM Lapl	GLM Lapl d	GLM VI	GLM VI d
australian	0.881 \pm 0.010	0.883 \pm 0.008	0.888 \pm 0.008	0.882 \pm 0.010	0.879 \pm 0.010	0.880 \pm 0.010	0.880 \pm 0.009
cancer	0.967 \pm 0.004	0.970 \pm 0.003	0.973 \pm 0.003	0.968 \pm 0.003	0.972 \pm 0.003	0.963 \pm 0.003	0.971 \pm 0.003
ionosphere	0.875 \pm 0.010	0.909 \pm 0.007	0.881 \pm 0.008	0.872 \pm 0.010	0.883 \pm 0.009	0.898 \pm 0.007	0.896 \pm 0.007
glass	0.706 \pm 0.011	0.672 \pm 0.015	0.550 \pm 0.014	0.691 \pm 0.010	0.691 \pm 0.013	0.700 \pm 0.013	0.669 \pm 0.013
vehicle	0.809 \pm 0.005	0.802 \pm 0.003	0.750 \pm 0.003	0.815 \pm 0.003	0.817 \pm 0.004	0.827 \pm 0.004	0.808 \pm 0.004
waveform	0.862 \pm 0.003	0.852 \pm 0.003	0.858 \pm 0.002	0.860 \pm 0.003	0.845 \pm 0.003	0.857 \pm 0.003	0.859 \pm 0.003
digits	0.977 \pm 0.001	0.969 \pm 0.001	0.954 \pm 0.001	0.971 \pm 0.001	0.963 \pm 0.001	0.966 \pm 0.001	0.967 \pm 0.001
satellite	0.919 \pm 0.001	0.900 \pm 0.001	0.863 \pm 0.001	0.918 \pm 0.001	0.917 \pm 0.001	0.918 \pm 0.001	0.912 \pm 0.001

Table B2. Test accuracy on UCI binary /multiclass classification tasks. 1 hidden layer, 50 units, \tanh activation. In comparison the the test log likelihoods (Tab. 1), BNN Laplace can achieve good performance on three data sets here. The neural network MAP performs better than on test log likelihoods. The proposed GLM VI approach achieves consistently good results.

Improving predictions of Bayesian neural networks via local linearization

Dataset	NN MAP	BBB	BNN Lapl	GLM Lapl	GLM Lapl d	GLM VI	GLM VI d
australian	0.063±0.005	0.061±0.003	0.144±0.007	0.061±0.006	0.082±0.006	0.055±0.005	0.058±0.003
cancer	0.035±0.003	0.032±0.003	0.086±0.004	0.034±0.003	0.048±0.003	0.026±0.002	0.027±0.002
ionosphere	0.080±0.006	0.083±0.002	0.214±0.006	0.081±0.004	0.132±0.007	0.077±0.004	0.067±0.005
glass	0.155±0.010	0.180±0.008	0.215±0.010	0.173±0.008	0.232±0.010	0.157±0.005	0.189±0.008
vehicle	0.078±0.005	0.060±0.002	0.255±0.003	0.088±0.003	0.181±0.005	0.065±0.003	0.070±0.003
waveform	0.044±0.002	0.064±0.002	0.160±0.002	0.055±0.001	0.078±0.003	0.052±0.002	0.049±0.002
digits	0.014±0.000	0.041±0.001	0.410±0.001	0.156±0.001	0.257±0.001	0.049±0.002	0.033±0.002
satellite	0.020±0.000	0.021±0.001	0.119±0.001	0.044±0.001	0.090±0.001	0.018±0.001	0.021±0.001

Table B3. Test ECE on UCI binary/multiclass classification tasks. 1 hidden layer, 50 units, tanh activation. The GLM VI method provides overall the best calibrated predictive probabilities according to the ECE metric. The MAP excels on the *digits* dataset.

Dataset	NN MAP	BBB	BNN Lapl	GLM Lapl	GLM Lapl d	GLM VI	GLM VI d
australian	0.31±0.01	0.34±0.01	0.42±0.00	0.32±0.02	0.33±0.01	0.32±0.02	0.31±0.01
cancer	0.11±0.02	0.11±0.01	0.19±0.00	0.10±0.01	0.11±0.01	0.11±0.01	0.12±0.02
ionosphere	0.35±0.02	0.41±0.01	0.50±0.00	0.29±0.01	0.35±0.01	0.35±0.05	0.32±0.03
glass	0.95±0.03	1.06±0.01	1.41±0.00	0.86±0.01	0.99±0.01	0.98±0.07	0.83±0.02
vehicle	0.420±0.007	0.504±0.006	0.885±0.002	0.428±0.005	0.618±0.003	0.402±0.007	0.432±0.005
waveform	0.335±0.004	0.393±0.003	0.516±0.002	0.339±0.004	0.388±0.003	0.335±0.004	0.364±0.008
digits	0.094±0.003	0.219±0.004	0.875±0.002	0.250±0.002	0.409±0.002	0.150±0.002	0.149±0.008
satellite	0.230±0.002	0.307±0.002	0.482±0.001	0.241±0.001	0.327±0.002	0.227±0.002	0.248±0.002

Table B4. Negative test log likelihood on UCI binary/multiclass classification tasks. 2 hidden layers, 50 units each, tanh activation. In comparison to a single hidden layer, BBB performs much worse in this benchmark. Further, MAP and GLM Laplace achieve slightly better performance compared to other approaches. Overall, test negative log likelihoods for a single layer are slightly better.

Dataset	NN MAP	BBB	BNN Lapl	GLM Lapl	GLM Lapl d	GLM VI	GLM VI d
australian	0.884±0.010	0.885±0.008	0.887±0.009	0.883±0.009	0.885±0.009	0.888±0.008	0.886±0.008
cancer	0.971±0.004	0.969±0.003	0.972±0.003	0.969±0.003	0.969±0.003	0.971±0.003	0.971±0.003
ionosphere	0.887±0.006	0.879±0.007	0.866±0.009	0.887±0.009	0.883±0.008	0.891±0.007	0.892±0.005
glass	0.684±0.010	0.534±0.013	0.459±0.009	0.678±0.013	0.666±0.013	0.675±0.015	0.669±0.012
vehicle	0.827±0.005	0.717±0.003	0.712±0.003	0.828±0.005	0.814±0.006	0.824±0.005	0.820±0.005
waveform	0.855±0.003	0.861±0.003	0.858±0.003	0.854±0.003	0.852±0.004	0.853±0.003	0.849±0.003
digits	0.974±0.001	0.951±0.002	0.924±0.001	0.964±0.001	0.960±0.001	0.966±0.001	0.970±0.001
satellite	0.916±0.001	0.891±0.001	0.842±0.001	0.915±0.001	0.910±0.001	0.917±0.001	0.911±0.001

Table B5. Test accuracy on UCI binary/multiclass classification tasks. 2 hidden layers, 50 units each, tanh activation. In comparison to test nll, many models achieve similar accuracies. GLM VI and MAP achieve consistently high performances.

Dataset	NN MAP	BBB	BNN Lapl	GLM Lapl	GLM Lapl d	GLM VI	GLM VI d
australian	0.056±0.004	0.099±0.005	0.185±0.008	0.055±0.005	0.094±0.007	0.060±0.005	0.050±0.003
cancer	0.033±0.002	0.033±0.002	0.127±0.003	0.033±0.002	0.053±0.003	0.027±0.003	0.027±0.003
ionosphere	0.092±0.004	0.155±0.009	0.238±0.008	0.089±0.003	0.131±0.006	0.081±0.004	0.081±0.004
glass	0.192±0.008	0.140±0.008	0.175±0.009	0.159±0.007	0.222±0.012	0.170±0.007	0.160±0.005
vehicle	0.064±0.003	0.058±0.002	0.266±0.003	0.078±0.003	0.210±0.005	0.052±0.002	0.071±0.002
waveform	0.041±0.002	0.134±0.002	0.233±0.003	0.049±0.002	0.109±0.004	0.038±0.001	0.049±0.002
digits	0.016±0.001	0.091±0.002	0.478±0.001	0.141±0.000	0.256±0.001	0.054±0.001	0.027±0.000
satellite	0.018±0.001	0.033±0.001	0.125±0.001	0.037±0.001	0.118±0.001	0.019±0.001	0.022±0.001

Table B6. Test ECE on UCI binary/multiclass classification tasks. 2 hidden layers, 50 units each, tanh activation. In line with the single layer results, the GLM VI methods are calibrated best.

Improving predictions of Bayesian neural networks via local linearization

Dataset	NN MAP	BBB	BNN Lapl	GLM Lapl	GLM Lapl d	GLM VI	GLM VI d
australian	0.32 \pm 0.01	0.32 \pm 0.01	0.38 \pm 0.01	0.32 \pm 0.01	0.33 \pm 0.01	0.32 \pm 0.01	0.31 \pm 0.01
cancer	0.13 \pm 0.03	0.17 \pm 0.05	0.14 \pm 0.01	0.10 \pm 0.01	0.12 \pm 0.01	0.16 \pm 0.05	0.17 \pm 0.05
ionosphere	0.28 \pm 0.02	0.23 \pm 0.02	0.43 \pm 0.01	0.28 \pm 0.01	0.34 \pm 0.01	0.25 \pm 0.02	0.35 \pm 0.08
glass	0.89 \pm 0.03	1.07 \pm 0.07	1.24 \pm 0.01	0.88 \pm 0.01	0.95 \pm 0.01	0.78 \pm 0.03	0.80 \pm 0.02
vehicle	0.387 \pm 0.008	0.367 \pm 0.006	0.756 \pm 0.003	0.460 \pm 0.004	0.524 \pm 0.004	0.386 \pm 0.007	0.417 \pm 0.006
waveform	0.363 \pm 0.004	0.354 \pm 0.005	0.438 \pm 0.002	0.368 \pm 0.004	0.379 \pm 0.003	0.377 \pm 0.006	0.356 \pm 0.005
digits	0.078 \pm 0.003	0.136 \pm 0.008	0.460 \pm 0.003	0.264 \pm 0.002	0.367 \pm 0.002	0.122 \pm 0.002	0.143 \pm 0.008
satellite	0.230 \pm 0.003	0.278 \pm 0.002	0.348 \pm 0.001	0.246 \pm 0.002	0.310 \pm 0.002	0.229 \pm 0.002	0.251 \pm 0.002

Table B7. Negative test log likelihood on UCI binary /multiclass classification tasks. 1 hidden layers, 50 units, relu activation. In comparison to relu activation, BBB improves over MAP and is close in performance to GLM VI methods. The best performance of the single layer tanh networks are slightly better than for relu presented here.

Dataset	NN MAP	BBB	BNN Lapl	GLM Lapl	GLM Lapl d	GLM VI	GLM VI d
australian	0.888 \pm 0.009	0.872 \pm 0.010	0.878 \pm 0.007	0.886 \pm 0.009	0.885 \pm 0.010	0.885 \pm 0.009	0.887 \pm 0.008
cancer	0.968 \pm 0.004	0.964 \pm 0.004	0.969 \pm 0.003	0.970 \pm 0.003	0.970 \pm 0.003	0.967 \pm 0.004	0.968 \pm 0.004
ionosphere	0.906 \pm 0.007	0.926 \pm 0.007	0.864 \pm 0.014	0.906 \pm 0.006	0.902 \pm 0.006	0.904 \pm 0.009	0.921 \pm 0.007
glass	0.672 \pm 0.014	0.634 \pm 0.014	0.562 \pm 0.011	0.653 \pm 0.014	0.659 \pm 0.013	0.678 \pm 0.014	0.684 \pm 0.015
vehicle	0.832 \pm 0.003	0.814 \pm 0.004	0.739 \pm 0.004	0.809 \pm 0.003	0.824 \pm 0.004	0.832 \pm 0.004	0.825 \pm 0.003
waveform	0.857 \pm 0.003	0.856 \pm 0.003	0.859 \pm 0.003	0.857 \pm 0.003	0.859 \pm 0.003	0.847 \pm 0.002	0.847 \pm 0.003
digits	0.981 \pm 0.001	0.973 \pm 0.001	0.952 \pm 0.001	0.971 \pm 0.001	0.957 \pm 0.001	0.969 \pm 0.001	0.976 \pm 0.001
satellite	0.916 \pm 0.001	0.899 \pm 0.001	0.892 \pm 0.001	0.913 \pm 0.001	0.908 \pm 0.002	0.912 \pm 0.001	0.908 \pm 0.001

Table B8. Test accuracy on UCI binary /multiclass classification tasks. 1 hidden layer, 50 units, relu activation. The MAP provides consistently the best accuracies on this architecture followed by the GLM VI methods.

Dataset	NN MAP	BBB	BNN Lapl	GLM Lapl	GLM Lapl d	GLM VI	GLM VI d
australian	0.056 \pm 0.004	0.099 \pm 0.005	0.185 \pm 0.008	0.055 \pm 0.005	0.094 \pm 0.007	0.060 \pm 0.005	0.050 \pm 0.003
cancer	0.033 \pm 0.002	0.033 \pm 0.002	0.127 \pm 0.003	0.033 \pm 0.002	0.053 \pm 0.003	0.027 \pm 0.003	0.027 \pm 0.003
ionosphere	0.092 \pm 0.004	0.155 \pm 0.009	0.238 \pm 0.008	0.089 \pm 0.003	0.131 \pm 0.006	0.081 \pm 0.004	0.081 \pm 0.004
glass	0.192 \pm 0.008	0.140 \pm 0.008	0.175 \pm 0.009	0.159 \pm 0.007	0.222 \pm 0.012	0.170 \pm 0.007	0.160 \pm 0.005
vehicle	0.064 \pm 0.003	0.058 \pm 0.002	0.266 \pm 0.003	0.078 \pm 0.003	0.210 \pm 0.005	0.052 \pm 0.002	0.071 \pm 0.002
waveform	0.041 \pm 0.002	0.134 \pm 0.002	0.233 \pm 0.003	0.049 \pm 0.002	0.109 \pm 0.004	0.038 \pm 0.001	0.049 \pm 0.002
digits	0.016 \pm 0.001	0.091 \pm 0.002	0.478 \pm 0.001	0.141 \pm 0.000	0.256 \pm 0.001	0.054 \pm 0.001	0.027 \pm 0.000
satellite	0.018 \pm 0.001	0.033 \pm 0.001	0.125 \pm 0.001	0.037 \pm 0.001	0.118 \pm 0.001	0.019 \pm 0.001	0.022 \pm 0.001

Table B9. Test ECE on UCI binary /multiclass classification tasks. 1 hidden layer, 50 units, relu activation. In line with the other architectures, the proposed GLM VI method provides the best calibrated predictive probabilities.

B.4. CIFAR

We use LeNet5 with 62K parameters and train on CIFAR10 dataset splitting the original training set (50K examples) into 10 splits of 5K examples to reduce complexity of inference in GGP and evaluate standard errors. We always test on the full original test set (10K examples). We tune the prior precision for i -th split using $(i + 1)$ -th split as validation dataset. We test 12 values for the prior precision uniformly spaced between 10 and 120. We train with Adam optimizer for 60 epochs. The initial learning rate is 10^{-3} and decayed by a factor of 10 after 35 and 50 epochs to ensure that we converge as close to θ_{MAP} as possible. We compare the performance of NN MAP (evaluated at θ^*), GGP Laplace evaluated at both θ^* and inferred θ_{MAP} and GGP VI with 512, 1024 and 2048 inducing points. We train for 200 epochs and minibatch size 512, decaying the learning rate by a factor of 0.97 at each epoch. We evaluate the performance of GLM and GGP methods by drawing 50,000 Monte Carlo samples from the diagonal GP predictive posterior, passing them through softmax function and estimating the expectation. We note that sampling from the functional posterior is very cheap as the functional posterior for a single data point is a C -dimensional diagonal Gaussian distribution.