Alexander Immer*, Maciej Korzepa, Matthias Bauer*

Improving predictions of Bayesian neural nets via local linearization (Appendix)

Contents of the Appendix

A   Derivations and additional details

In this section we provide additional derivations and details on the background as well as proposed methods, the GLM predictive and its GP equivalent. As discussed in the main paper, the Laplace-GGN approximation to the posterior at the MAP, $\boldsymbol{\theta}^* = \boldsymbol{\theta}_{\mathrm{MAP}}$, takes the form:

$$q_{\mathrm{GGN}}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}_{\mathrm{MAP}}, \mathbf{S}) = \mathcal{N}\left(\boldsymbol{\theta}_{\mathrm{MAP}}, \left(\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x})^{\mathsf{T}} \mathbf{\Lambda}(\mathbf{y};\mathbf{f}) \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x}) + \mathbf{S}_0^{-1}\right)^{-1}\right), \tag{12}$$

where $\mathbf{S}_0$ denotes the prior covariance of the parameters, $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{S}_0)$. Because using the full posterior covariance is infeasible in practise, we have to apply further approximations. In App. A.1 we discuss the KFAC approximation that is both expressive and scalable and therefore our choice for large-scale experiments. In App. A.2, we discuss GP inference with special attention to the subset-of-data approach. At the end of this section, we discuss the computational complexities of the proposed predictives and compare them to the BNN predictive.

A.1   KFAC approximation of the Laplace-GGN

Ritter et al. (2018) first proposed a Laplace-GGN approximation that utilizes a KFAC approximation to the posterior enabled by scalable KFAC approximations to the GGN and Fisher information matrix (Botev et al., 2017; Martens and Grosse, 2015). Here, we discuss the KFAC posterior approximation that we use and compare it to theirs. In particular, we do not use nor require dampening and post-hoc adjustment of the prior precision which would render the MAP estimate invalid. We use the KFAC posterior approximation for large-scale image classification tasks because it can model parameter covariances per layer while maintaining scalability; a diagonal posterior approximation fails to model these relationships while the full GGN is intractable for larger networks.

Botev et al. (2017) and Martens and Grosse (2015) propose a KFAC approximation to the expected GGN Hessian approximation under the data distribution and (for the likelihoods considered) Fisher information matrix, respectively. The KFAC approximation typically is block-diagonal and factorizes across neural network layers. Critically, the KFAC approximation is based on the observation that the GGN approximation to the Hessian

of the likelihood for *a single data point* can be written as a Kronecker product exactly. For a single data point $(\mathbf{x}_n, \mathbf{y}_n)$ and parameter group l, we can write

$$\left[\mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x}_n)^{\mathsf{T}} \boldsymbol{\Lambda}(\mathbf{y}_n; \mathbf{f}_n) \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x}_n)\right]_l = \mathbf{Q}_l^{(n)} \otimes \mathbf{W}_l^{(n)}, \tag{A.1}$$

where $\mathbf{Q}_l^{(n)}$ depends on the pre-activations of layer l and $\mathbf{W}_l^{(n)}$ can be computed recursively; for details, see Botev et al. (2017). In the case of a fully connected layer parameterized by $\mathbf{W} \in \mathbb{R}^{D_{\text{out}}^{(l)} \times D_{\text{in}}^{(l)}}$, we map from an internal representation of dimensionality $D_{\text{in}}^{(l)}$ to dimensionality $D_{\text{out}}^{(l)}$. Then, the Kronecker factors in Eq. (A.1) are square matrices: $\mathbf{Q}_l^{(n)} \in \mathbb{R}^{D_{\text{out}}^{(l)} \times D_{\text{out}}^{(l)}}$ and $\mathbf{W}_l^{(n)} \in \mathbb{R}^{D_{\text{in}}^{(l)} \times D_{\text{in}}^{(l)}}$. Critically, both matrices are positive semi-definite.

However, the full GGN requires a sum of Eq. (A.1) over all data points. While each individual summand allows a Kronecker factorization, this is not necessarily the case for the sum. For tractability, Martens and Grosse (2015) and Botev et al. (2017) propose to approximate the sum of Kronecker products by a Kronecker product of sums:

$$\left[\sum_{n=1}^{N} \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x}_n)^{\mathsf{T}} \boldsymbol{\Lambda}(\mathbf{y}_n; \mathbf{f}_n) \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x}_n)\right]_l = \sum_{n=1}^{N} \mathbf{Q}_l^{(n)} \otimes \mathbf{W}_l^{(n)} \approx \left(\sum_{n=1}^{N} \mathbf{Q}_l^{(n)}\right) \otimes \left(\sum_{n=1}^{N} \mathbf{W}_l^{(n)}\right) =: \mathbf{Q}_l \otimes \mathbf{W}_l, \tag{A.2}$$

which defines $\mathbf{Q}_l$ and $\mathbf{W}_l$ as the sum of per-data-point Kronecker factors. Approximating the sum of products with a product of sums is quite crude since we add additional cross-terms due to the distributivity of the Kronecker product.

The KFAC Laplace-GGN posterior approximation is given by using the above approximation to the GGN in the definition of the posterior covariance $\mathbf{S}$ in Eq. (12). As considered in the experiments and typical for Bayesian deep learning (Ritter et al., 2018; Khan et al., 2018; Khan et al., 2019; Zhang et al., 2018), we use an isotropic Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(0, \delta^{-1}\boldsymbol{I})$. We denote by $\mathbf{S}_{\text{KFAC}} \approx \mathbf{S}$ the KFAC Laplace-GGN approximation which is then given by

$$\mathbf{S}_{\text{KFAC}}^{-1} = \mathbf{Q}_l \otimes \mathbf{W}_l + \delta\boldsymbol{I}_l, \tag{A.3}$$

where $\boldsymbol{I}_l$ is the identity matrix with the number of parameters of the l-th layer. The entire term does not necessarily allow a Kronecker-factored representation.

To that end, Ritter et al. (2018) propose to approximate the posterior precision further, motivated by maintaining a Kronecker factored form, as:

$$\mathbf{S}_{\text{KFAC}}^{-1} \approx (\mathbf{Q}_l + \sqrt{\delta}\boldsymbol{I}) \otimes (\mathbf{W}_l + \sqrt{\delta}\boldsymbol{I}) = \mathbf{Q}_l \otimes \mathbf{W}_l + \mathbf{Q}_l \otimes \sqrt{\delta}\boldsymbol{I} + \sqrt{\delta}\boldsymbol{I} \otimes \mathbf{W}_l + \delta\boldsymbol{I}_l, \tag{A.4}$$

which artificially increases the posterior concentration by the cross-products. That is, because the cross-products are positive semi-definite matrices, they increase the eigenvalues of the posterior precision and thereby reduce the posterior covariance, see Fig. B.2 for a visual example. This further approximation is commonly referred to as *dampening* in the context of second-order optimization methods (Botev et al., 2017; Zhang et al., 2018). While Ritter et al. (2018) do not motivate this approximation in particular, we find that it is in fact necessary to mediate underfitting issues of BNN predictive when using a Laplace-GGN posterior, see Sec. 4.3 and App. B.4. Our proposed GLM predictive does not require a dampened/concentrated posterior, but also works in this case. We now show that the dampening approximation in Eq. (A.4) is, in fact, also not necessary from a computational perspective; that is, we show how to compute $\mathbf{S}_{\text{KFAC}}$ (Eq. (A.3)) without this approximation while keeping the complexity unchanged.

To avoid the additional approximation in form of *dampening* that artificially concentrates the posterior, we work with an eigendecomposition of the Kronecker factors. Let $\mathbf{Q}_l = \mathbf{M}_{Q_l} \mathbf{D}_{Q_l} \mathbf{M}_{Q_l}^{\mathsf{T}}$ be the eigendecomposition of $\mathbf{Q}_l$ and likewise for $\mathbf{W}_l$, where $\mathbf{D}$ is a diagonal matrix of eigenvalues that are non-negative as both Kronecker factors are positive semi-definite. We have

$$\begin{aligned}
\mathbf{S}_{\text{KFAC}}^{-1} &= (\mathbf{M}_{Q_l}\mathbf{D}_{Q_l}\mathbf{M}_{Q_l}^{\mathsf{T}}) \otimes (\mathbf{M}_{W_l}\mathbf{D}_{W_l}\mathbf{M}_{W_l}^{\mathsf{T}}) + \delta\boldsymbol{I}_l \\
&= (\mathbf{M}_{Q_l} \otimes \mathbf{M}_{W_l})(\mathbf{D}_{Q_l} \otimes \mathbf{D}_{W_l})(\mathbf{M}_{Q_l}^{\mathsf{T}} \otimes \mathbf{M}_{W_l}^{\mathsf{T}}) + \delta(\mathbf{M}_{Q_l} \otimes \mathbf{M}_{W_l})(\mathbf{M}_{Q_l}^{\mathsf{T}} \otimes \mathbf{M}_{W_l}^{\mathsf{T}}) \\
&= (\mathbf{M}_{Q_l} \otimes \mathbf{M}_{W_l})(\mathbf{D}_{Q_l} \otimes \mathbf{D}_{W_l} + \delta\boldsymbol{I}_l)(\mathbf{M}_{Q_l}^{\mathsf{T}} \otimes \mathbf{M}_{W_l}^{\mathsf{T}}),
\end{aligned}$$

which follows form the fact that $\mathbf{M}_{Q_l} \otimes \mathbf{M}_{W_l}$ forms the eigenvectors of $\mathbf{Q} \otimes \mathbf{W}$ and is therefore unitary and can be used to write the identity $\boldsymbol{I}_l$.

Based on the eigendecomposition of the individual Kronecker factors, the required quantities for sampling from the posterior (square root) or inversion can be computed efficiently. In comparison to the approximation of Ritter et al. (2018), we only incur additional storage costs due to the eigenvalues $\mathbf{D}$ which are negligible in size since they are diagonal and the major cost drivers are the dense matrices $\mathbf{M}$.

Working with the eigendecomposition further enables to understand the posterior concentration induced by dampening better. Recall that $\mathbf{S}_{\mathrm{KFAC}}^{-1}$ is the posterior parameter precision and therefore higher concentration (i.e. eigenvalues) lead to less posterior uncertainty. instead of the diagonal $\mathbf{D}_{Q_l} \otimes \mathbf{D}_{W_l} + \delta \boldsymbol{I}_l$, the approximation of Ritter et al. (2018) *adds* the additional diagonal terms $\mathbf{D}_{Q_l} \otimes \sqrt{\delta}\boldsymbol{I} + \sqrt{\delta}\boldsymbol{I} \otimes \mathbf{D}_{W_l}$. As all eigenvalues are positive and $\delta > 0$, this corresponds to an artificial concentration of the posterior by increasing the eigenvalues. The concentration is also uncontrolled as we scale the eigenvalues of the Kronecker factors by the prior precision – normally, these terms should only be combined additively.

A.2   Gaussian process approximate inference

In Sec. 3.5 we presented an equivalent Gaussian process (GP) formulation of the GLM, which can give rise to orthogonal posterior approximations.

The starting point is the Bayesian GLM with linearized features

$$\mathbf{f}_{\mathrm{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{f}(\mathbf{x}, \boldsymbol{\theta}^*) + \mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x})(\boldsymbol{\theta} - \boldsymbol{\theta}^*), \tag{10}$$

and prior on the parameters $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{m}_0, \mathbf{S}_0)$. As also discussed in the main paper, we obtain the corresponding Gaussian process prior model with mean function $\mathbf{m}(\mathbf{x})$ and covariance (kernel) function $\mathbf{k}(\mathbf{x}, \mathbf{x}')$ by computing the mean and covariance of the features (Rasmussen and Williams, 2006):

$$\begin{aligned} \mathbf{m}(\mathbf{x}) &= \mathbb{E}_{p(\boldsymbol{\theta})}[\mathbf{f}_{\mathrm{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}; \boldsymbol{\theta})] = \mathbf{f}_{\mathrm{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}; \mathbf{m}_0) \\ \mathbf{k}(\mathbf{x}, \mathbf{x}') &= \mathrm{Cov}_{p(\boldsymbol{\theta})}[\mathbf{f}_{\mathrm{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}; \boldsymbol{\theta}), \mathbf{f}_{\mathrm{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}'; \boldsymbol{\theta})] = \mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x})\mathbf{S}_0\mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x}')^{\mathsf{T}}. \end{aligned} \tag{14}$$

This specifies the functional GP prior $p(\mathbf{f})$ over the C-dimensional function outputs, which is then used together with the same likelihood as for the GLM. In line with the BNN and GLM model, we choose an isotropic prior on the parameters with covariance $\mathbf{S}_0 = \delta^{-1}\boldsymbol{I}$ and zero mean $\mathbf{m}_0 = 0$. The GP mean function is therefore given by $\mathbf{m}(\mathbf{x}) = \mathbf{f}(\mathbf{x}, \boldsymbol{\theta}^*) - \mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x})\boldsymbol{\theta}^*)$, such that the latent function $\mathbf{f}(X)$ is modelled by this mean function and a zero centred GP that models the fluctuations around the mean. For multiple outputs (e.g. multi-class classification), we assume independent GPs per output dimension.

The GP log joint distribution can be written as (see Sec. 3.5)

$$\log p(\mathcal{D}, \mathbf{f}(X)) = \sum_{n=1}^{N} \log p(\mathbf{y}_n | \mathbf{f}_n) + \log p(\mathbf{f}). \tag{A.5}$$

We now need to compute the functional posterior distribution over the training data set $\mathbf{f}(X) \in \mathbb{R}^{NC}$, which does not have a closed form for general likelihoods.

A.2.1   Laplace approximation

In our work, we consider the Laplace approximation to the GP posterior (Rasmussen and Williams, 2006) under the assumption that $\boldsymbol{\theta}^* = \boldsymbol{\theta}_{\mathrm{MAP}}$. Here, we present details on the Laplace approximation and give formulas for the multi-output case with C outputs and N data points. We follow Pan et al. (2020) and assume that the mode of the GLM and GP coincide, which allows us to directly construct a Laplace approximation at $\mathbf{f}_{\mathrm{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}; \boldsymbol{\theta}_{\mathrm{MAP}})$.

The posterior predictive covariance is given by the Hessian of the log joint distribution Eq. (A.5) w.r.t the latent function $\mathbf{f}$, $\nabla_{\mathbf{f}\mathbf{f}}^2 \log p(\mathcal{D}, \mathbf{f}(X))$, similarly to in the parametric Laplace approximation. For multi-output predictions, we obtain a block diagonal covariance matrix, because we assume independent GPs for each output. To define the multi-output predictive covariance, we therefore only need to compute the additional stacked second derivative of the log likelihood, the $\mathbf{\Lambda}(\mathbf{y}; \mathbf{f})$ terms, in the right format. We define the block-diagonal matrix $\boldsymbol{L}_{NN} \in \mathbb{R}^{NC \times NC}$ where the n-th block is given by the $C \times C$ matrix $\mathbf{\Lambda}(\mathbf{y}_n, \mathbf{f}(\mathbf{x}_n, \boldsymbol{\theta}^*))$. We then obtain

$\nabla^2_{\mathbf{ff}} \log p(\mathcal{D}, \mathbf{f}(X)) = -\boldsymbol{L}_{NN} - \delta(\mathcal{J}_{\boldsymbol{\theta}^*}(X)\mathcal{J}_{\boldsymbol{\theta}^*}(X)^\mathsf{T})^{-1}$ similar to Rasmussen and Williams (2006, chap. 3). Using the matrix inversion Lemma, we can then write the posterior as

$$q(\mathbf{f}^*|\mathbf{x}^*, \mathcal{D}) = \mathcal{N}\left(\mathbf{f}(\mathbf{x}^*; \boldsymbol{\theta}^*), \boldsymbol{\Sigma}_*\right) \quad \text{with} \quad \boldsymbol{\Sigma}_* = \boldsymbol{K}_{**} - \boldsymbol{K}_{*N}(\boldsymbol{K}_{NN} + \boldsymbol{L}_{NN}^{-1})^{-1}\boldsymbol{K}_{N*}, \tag{A.6}$$

where $\boldsymbol{K}_{*N}$ is the kernel between the input location and the N training samples, i.e., $\boldsymbol{K}_{*N} = \delta^{-1}\mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x}^*)\mathcal{J}_{\boldsymbol{\theta}^*}(X)^\mathsf{T}$. In comparison to Eq. (15) for a single output, the kernel evaluated on the training data now has size $NC \times NC$ instead of $N \times N$. Note that the predictive mean is simply defined by the MAP of the neural network, the GP simply augments it with predictive uncertainty. Assuming independent prior GPs for each output, i.e., $p(\mathbf{f}) = \prod_{c=1}^C p(\mathbf{f}_c)$, scalability can be maintained due to parallelization as we can deal with C $N \times N$ kernels independently in parallel (Rasmussen and Williams, 2006).

A.2.2 Subset of data

However, even $N \times N$ kernel matrices are intractable for larger datasets ($\mathcal{O}(N^2)$ memory and $\mathcal{O}(N^3)$ inversion cost); we therefore have to resort to further approximations. In this work we arguably use the simplest such approach, subset-of-data (SoD), in which we only retain M of the original N training points for inference. Depending on the dataset, already a relatively small number of data points can yield reasonable results. In practise, we choose M as large as computationally feasible, and obtain the following posterior covariance approximation:

$$\boldsymbol{\Sigma}_{*,\text{SoD}} = \boldsymbol{K}_{**} - \boldsymbol{K}_{*M}(\boldsymbol{K}_{MM} + \boldsymbol{L}_{MM}^{-1})^{-1}\boldsymbol{K}_{M*}. \tag{A.7}$$

We further scale the prior precision δ, which is part of the GP kernel, by a factor proportional to $\frac{N}{M}$ to alleviate problems due to using a small subset of the data. If we only use a subset of the data, the predictive variance is not sufficiently reduced due to the subtracted gain in Eq. (A.7). Reducing the GP prior covariance by a factor allows to alleviate this effect. The factor is selected on the held-out validation set in our experiments but $\frac{N}{M}$ is a good rule of thumb that always gives good performance.

Instead of a subset-of-data approach, we could use more elaborate sparse Gaussian process approximations (Titsias, 2009; Hensman et al., 2015); however, we found that a subset-of-data worked relatively well, such that we leave other sparse approximations for future work.

A.3 Computational considerations and complexities

We discuss the computational complexities of the proposed and related methods in detail in this section. The central quantity of interest for the GGN as well as the GP and GLM predictives are the Jacobians. As discussed in Sec. 3.6, computation and storage of Jacobians for typical network architectures is $\mathcal{O}(PC)$ for a single data point. We first discuss computation, storage, and inversion of various Laplace-GGN posterior approximations and after discuss the GLM and GP inference (refinement) and predictives.

A.3.1 Laplace-GGN

Apart from computation of the Laplace-GGN posterior precision in Eq. (12), we also need to invert it and decompose it (for sampling) and store it. The full covariance Laplace-GGN posterior approximation is very costly as it models the covariance between all parameters in the neural network: we need to compute the sum and matrix multiplications in Eq. (12) resulting in $\mathcal{O}(NC^2P^2)$ complexity. Storing this matrix and decomposing it is in $\mathcal{O}(P^2)$ and $\mathcal{O}(P^3)$, respectively. Therefore, it can only be applied to problems where the neural network has $\mathcal{O}(10^4)$ parameters.

A cheap alternative is to use the diagonal GGN approximation which can be computed exactly in $\mathcal{O}(NC^2P)$ or approximately (using sampling) in $\mathcal{O}(NCP)$ (Dangel et al., 2019). It is also considered in iterative algorithms for variational inference, e.g. by Khan et al. (2018) and Osawa et al. (2019) Storage, decomposition, and inversion of the diagonal is as cheap as storing one set of parameter with $\mathcal{O}(P)$.

For the KFAC posterior approximation, the network architecture decides the complexity of inference: for simplicity, we assume a fully connected network but the analysis can be readily extended to convolutional layers or architectures. For layer l that maps from a $D_{\text{in}}^{(l)}$ to a $D_{\text{out}}^{(l)}$-dimensional representation, the corresponding block in the GGN approximation is Kronecker factored, i.e., with $\mathbf{Q}_l \otimes \mathbf{W}_l$ (see Eq. (8)) where $\mathbf{Q}_l \in \mathbb{R}^{D_{\text{in}}^{(l)} \times D_{\text{in}}^{(l)}}$ and $\mathbf{W}_l \in \mathbb{R}^{D_{\text{out}}^{(l)} \times D_{\text{out}}^{(l)}}$.

As shown in App. A.1, all relevant quantities can be computed by decomposing both Kronecker factors individually due to the properties of the Kronecker product. Therefore, storage complexity is simply $\mathcal{O}\left(\sum_{l=1}^{L}(D_{\text{in}}^{(l)})^2 + (D_{\text{out}}^{(l)})^2\right)$ and decomposition $\mathcal{O}\left(\sum_{l=1}^{L}(D_{\text{in}}^{(l)})^3 + (D_{\text{out}}^{(l)})^3\right)$. Suppose $D_l := D_{\text{in}}^{(l)} = D_{\text{in}}^{(l)}$, i.e., same in- as output dimensionality of the layer, then the KFAC approximation avoids having to handle a matrix with D_l^4 entries by instead handling two D_l^2 matrices (Botev et al., 2017). Therefore, KFAC approximations are our default choice for larger scale inference on the parametric side.

Above, we discussed Laplace-GGN approximation using the full training data set. We can also obtain a stochastic unbiased estimate of the GGN approximation to the Hessian by subsampling M data points and scaling the GGN by factor $\frac{N}{M}$ which is used in stochastic variational inference (Khan et al., 2018; Osawa et al., 2019; Zhang et al., 2018) and leads to a reduction in complexity where we have factor M instead of N and $M \ll N$.

The cost of the functional Laplace-GGN approximation in the equivalent GP model is dominated by the number of data points N instead of the number of parameters P. To obtain the full functional posterior approximation is in $\mathcal{O}\left(CN^3 + CN^2P\right)$ due to C independent GPs (Rasmussen and Williams, 2006) and the computation of the N^2 inner products of Jacobians. The proposed subset-of-data approximation of the kernel matrix simply leads to replacement of N with $M \ll N$ in the above complexities and makes inference tractable for large data sets.

A.3.2   GLM and GP refinement

To refine the GLM, we need to update the parameter vector $\boldsymbol{\theta} \in \mathbb{R}^P$. For example, we can use gradient-based optimization to obtain the MAP of the GLM model or alternatively construct a variational or sampling-based posterior approximation. In any case, we need to evaluate the density or gradients which require access to a mini-batch of $N_{\text{bs}} \leq N$ Jacobians; $N_{\text{bs}} \ll N$ if we can use stochastic estimators. Hence, one step of refinement is at least in $\mathcal{O}\left(N_{\text{bs}}PC\right)$. In contrast, one optimization step of the corresponding neural network would be $\mathcal{O}\left(N_{\text{bs}}P\right)$ instead. While the complexity is only higher than for basic neural network training by factor C, ad-hoc computation of Jacobians is practically much slower in current deep learning frameworks. An alternative is to pre-compute and cache Jacobians which is feasible on small data sets (Sec. 4.2 and B.3) but leads to memory problems on larger architectures. Therefore, we only consider it on the smaller examples.

To refine the GP, we need to update the function vector $\mathbf{f}(X) \in \mathbb{R}^{MC}$ (see App. A.2) where $M \leq N$ in case of a subset approximation. For GP refinement, we need to compute the local linearization (Eq. (10)) up front which is $\mathcal{O}\left(MPC\right)$. Following steps, e.g., gradient computations, are then typically $\mathcal{O}\left(MC\right)$. Depending on P, M, N_{bs}, either parametric or functional refinement may be preferred.

A.3.3   Predictives

We discuss the cost of obtaining samples from the posterior predictive using the proposed GLM and GP predictives and compare them to the BNN predictive. First, we describe the complexity and procedure of naive parameter-level sampling and then describe how more samples can be obtained cheaply using local reparameterization (Kingma et al., 2015). Local reparameterization allows us to sample the outputs of a linear model directly instead of sampling the (typically much higher dimensional) weights. The alternative GP predictive immediately produces samples from the output as well but has a different complexity.

The basic form of GLM and BNN predictives relies on parameters sampled from the Laplace-GGN posterior and sampling depends directly on the sparsity of posterior approximation (DIAG, KFAC, full). Here, we assume we have already inverted and decomposed the posterior precision (see App. A.3.1) which enables sampling. Evaluating the predictive for a single sample is $\mathcal{O}\left(P\right)$ for the BNN and $\mathcal{O}\left(PC\right)$ for the GLM. Typically, however, the major cost is due to sampling parameters: sampling once from a full covariance is $\mathcal{O}\left(P^2\right)$ for, from a diagonal $\mathcal{O}\left(P\right)$, and from the KFAC posterior $\mathcal{O}\left(\sum_{l=1}^{L} D_{\text{out}}^{(l)}(D_{\text{in}}^{(l)})^2 + D_{\text{in}}^{(l)}(D_{\text{out}}^{(l)})^2\right)$. The complexity of sampling from the KFAC posterior is a consequence of sampling from the matrix normal distribution (Gupta and Nagar, 1999).

For both the GLM predictive and the BNN predictive, we can make use of local reparameterization (Kingma et al., 2015) to speed up the sampling. Local reparameterization uses the fact sums of Gaussian random variables are themselves Gaussian random variables – instead of sampling Gaussian weights individually, we can sample their sum (the outputs) directly. For the linear GLM, we can do this for all covariances of the posterior, whereas for BNNs this is only effective for diagonal covariance posteriors, as we describe in the following.

For the BNN, we can use local reparameterization to sample vectors of smaller size from the hidden representations instead of the parameter matrices (Kingma et al., 2015). However, this is only possible when we have a diagonal posterior approximation where sampling a parameter is of same complexity as a forward pass $\mathcal{O}\left(P\right)$. Here, we consider mostly KFAC and full posterior approximations where we need to fall back to the naive procedure of sampling parameters. For the GLM, we can directly reparameterize into the C-dimensional output distribution by mapping from the distribution on $\boldsymbol{\theta}$ to a distribution on $\mathcal{J}_{\boldsymbol{\theta}^*}(\mathbf{x}^*)\boldsymbol{\theta}$. The complexity of this transformation again depends on the sparsity of the Laplace-GGN approximation. We have for the full covariance $\mathcal{O}\left(C^2P + CP^2\right)$, for diagonal $\mathcal{O}\left(C^2P\right)$, and for KFAC $\mathcal{O}\left(C^2P + \sum_{l=1}^{L} D_{\text{out}}^{(l)}(D_{\text{in}}^{(l)})^2 + D_{\text{in}}^{(l)}(D_{\text{out}}^{(l)})^2\right)$. The transformation results in a multivariate Normal distribution on the C outputs. The additional C^2 terms arise because we obtain a $C \times C$ functional output covariance and can be reduced to $\mathcal{O}\left(C\right)$ by using only a diagonal approximation to it. Using a full multivariate normal additionally requires decomposition of the $C \times C$ covariance in $\mathcal{O}\left(C^3\right)$ while the diagonal approximation incurs no additional cost to sample outputs. Critically, having a distribution on the C outputs allows much cheaper sampling in many cases. For example, consider a full Laplace-GGN posterior approximation: naive sampling in the GLM with S predictive samples is $\mathcal{O}\left(SP^2 + SPC\right)$. In contrast, transforming and sampling S samples directly using a diagonal output distribution is $\mathcal{O}\left(CP^2 + SC\right)$. Therefore, it scales better in the number of samples and is always more efficient once we sample more outputs than we have classes which is realistic.

To predict with the GP posterior approximation using a subset of $M \leq N$ training points, we need to compute the GP predictive in Eq. (16). The output distribution (before the inverse link $g(\cdot)$), in particular its covariance, can therefore be obtained in $\mathcal{O}\left(CMP + CM^2\right)$ for C independent outputs; the first term arises from computing the kernel between the test point and the M training points and the second term from computing the gain term in the predictive. Sampling from the GP predictive is $\mathcal{O}\left(SC\right)$ since we sample outputs from a diagonal Gaussian; in the regression case with a Gaussian likelihood, this additional step is not necessary.

B   Experimental details and additional results

B.1   Illustrative $1d$ example

In Sec. 3.4 we considered a very simple $1d$ binary classification problem with inputs $\mathbf{x}_i$ given by $\{-6, -4, -2, 2, 4, 6\}$ and corresponding labels $\mathbf{y}_i$ given by 0 for $\mathbf{x}_i < 0$ and 1 for $\mathbf{x}_i > 0$ (see Fig. B.1 *left*). This dataset has ambiguity about the exact location of the decision boundary. We use a Bernoulli likelihood, sigmoid inverse link function and features $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}) = 5 \tanh(\mathbf{wx} + \mathbf{b})$. The model parameters $\boldsymbol{\theta}$ are given by $\boldsymbol{\theta} = \{\mathbf{w}, \mathbf{b}\}$, $\mathbf{w}, \mathbf{b}, \mathbf{x} \in \mathbb{R}$, and we use a factorized Gaussian prior $p(\mathbf{w}, \mathbf{b}) = \mathcal{N}(\mathbf{w}; 0, 1)\mathcal{N}(\mathbf{b}; 0, 1)$. Because there is ambiguity in the data, we cannot determine the parameter values with certainty, such that a Bayesian treatment is adequate.

For this low dimensional problem, we can find the "true" posterior $p(\boldsymbol{\theta}|\mathcal{D})$ through HMC sampling (Neal, 2010), see Fig. B.1 *(left)*, which is symmetric w.r.t the line $\mathbf{b} = 0$ but skewed along the $\mathbf{w}$ axis and concentrated on values $\mathbf{w} > 0$.

Using GD we find the MAP estimate $\boldsymbol{\theta}_{\mathrm{MAP}} = (\mathbf{w}_{\mathrm{MAP}}, \mathbf{b}_{\mathrm{MAP}})$ and employ the full Laplace approximation to find an approximate posterior, $q_{\mathrm{Lap}}(\boldsymbol{\theta})$. We also compute the GGN approximation at $\boldsymbol{\theta}^* = \boldsymbol{\theta}_{\mathrm{MAP}}$ and compute the corresponding Laplace-GGN posterior, $q_{\mathrm{GGN}}(\boldsymbol{\theta})$, as well as the dampened Laplace-GGN posterior as proposed by Ritter et al. (2018), $q_{\mathrm{Ritter}}(\boldsymbol{\theta})$. As they are Gaussian approximations, they are unskewed along the $\mathbf{w}$ axis but symmetric w.r.t $\mathbf{b} = \mathbf{b}_{\mathrm{MAP}} = 0$ and $\mathbf{w} = \mathbf{w}_{\mathrm{MAP}}$. Notably, they all put probability mass into the region $\mathbf{w} < 0$. In stark contrast, the true posterior (HMC) does not put any probability mass in this region. We find that the Laplace-GGN posterior is less concentrated than the full Laplace posterior; the dampened posterior by Ritter et al. (2018) is artificially concentrated and even narrower than the full Laplace posterior due to dampening. See Fig. B.2 for a contour plot of all posteriors considered.

We can compute the posterior distribution over the preactivations $\mathbf{f}$, $p(\mathbf{f}|\mathbf{x})$, as well as the posterior predictive for the class label $\mathbf{y}$, $p(\mathbf{y}|\mathbf{f}(\mathbf{x}))$. We use the BNN predictive on all posteriors and evaluate the proposed GLM predictive on the Laplace-GGN posterior. The GLM predictive uses linearized features $\mathbf{f}_{\mathrm{lin}}^{\boldsymbol{\theta}^*}$ in the predictive and we show these linearized features for $p(\mathbf{f}|\mathbf{x})$ in this case, see Fig. B.2 *(middle row)* for the preactivations and Fig. B.2 *(bottom row)* for the predictives. We also show the preactivations and predictives for HMC and the Laplace-GGN posterior (together with their feature functions) in Fig. B.1.

While the true (HMC) posterior with the corresponding BNN predictive yields reasonable uncertainties, all Laplace posteriors using the BNN predictive underfit or severely underfit; this is due to the posterior probability density having mass in $\mathbf{w} < 0$, which means that we obtain a bimodal predictive distribution. The dampened posterior (Ritter et al., 2018) has the most concentrated posterior, such that it suffers the least. In stark contrast, the proposed GLM predictive despite using the widest posterior produces reasonable preactivations and a posterior predictive that does not underfit and is much closer to the true (HMC) predictive. The median predictions for all methods (—— in Fig. B.2) saturate at $\mathbf{y} \approx 0$ or $\mathbf{y} \approx 1$, whereas this is only true for the mean (--- in Fig. B.2) when using HMC or the proposed GLM predictive.

In Fig. B.1 *(right)* we show the posterior predictive densities at a specific input location $\mathbf{x} = 3$. While the HMC and GLM predictive have a single mode at $\mathbf{y} = 1$, the BNN predictive underfits and is, in fact, bimodal with a mode at $\mathbf{y} = 1$ and a second mode $\mathbf{y} = 0$. This second underfitting mode is due to posterior samples from the mismatched region with $w < 0$ as also discussed in the main text.
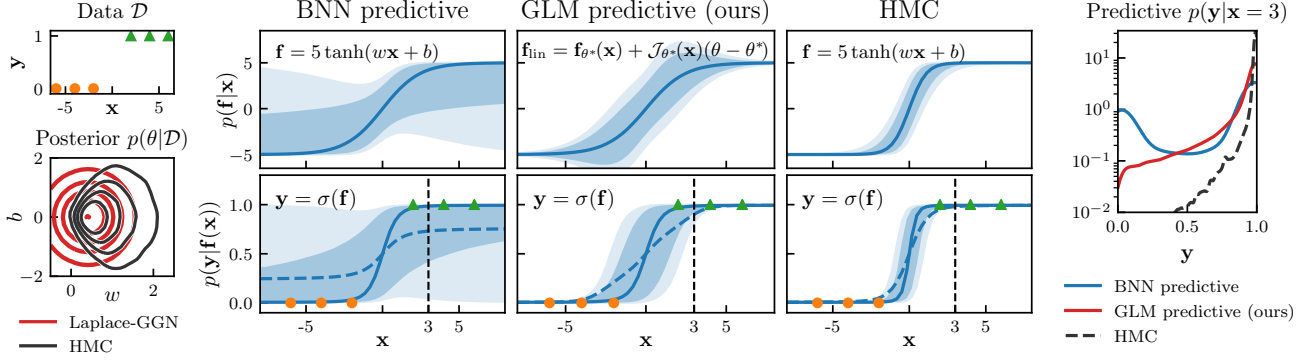
Figure B.1: Illustration how the GLM predictive overcomes underfitting of the BNN predictive and closely resembles the true predictive (HMC). $1d$ binary classification ($\bullet$ vs $\blacktriangle$) with a 2-parameter feature function $\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}) = 5\texttt{tanh}(w\mathbf{x} + b)$, $\boldsymbol{\theta} = (w, b)$, classification likelihood $p(\mathbf{y} = 1|\mathbf{f}) = \sigma(\mathbf{f})$, and prior $p(\boldsymbol{\theta}_i) = \mathcal{N}(0, 1)$.

left: Laplace-GGN posterior around $\boldsymbol{\theta}_{\text{MAP}}$ vs. the true posterior (10^5 HMC samples): the Laplace-GGN is symmetric and extends beyond the true, skewed posterior with same MAP.

middle:. The GLM predictive with GGN posterior yields reasonable linearized preactivations $\mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}$ and its predictive $p(\mathbf{y}|\mathbf{f}(\mathbf{x}))$ closely resembles the true predictive (HMC); the BNN predictive underfits. ▮ innermost 50%/66% of samples; — median, --- mean.

right: Predictive density $p(\mathbf{y}|\mathbf{x})$ at $\mathbf{x} = 3$: the BNN predictive is bimodal due to the behaviour of the tanh for posterior samples with positive $w > 0$ or negative $w < 0$ slopes.
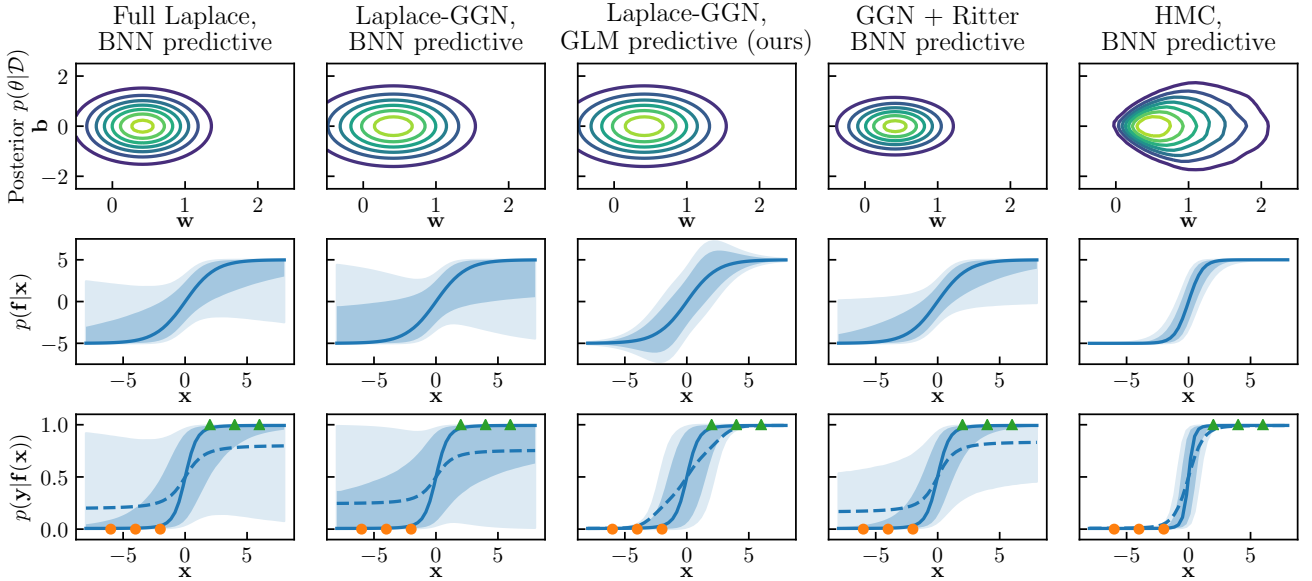
Figure B.2: Continuation of Fig. B.1.

Left to right: The full Laplace posterior with BNN predictive, the Laplace-GGN posterior with BNN as well as GLM predictive, the dampened Laplace-GGN posterior with BNN predictive by Ritter et al. (2018), and the true (HMC) posterior (10^5 samples) with BNN predictive.

top: Parameter posteriors or posterior approximations. The Laplace posteriors are symmetric and extends beyond the skewed true (HMC) posterior with same MAP, $\boldsymbol{\theta}_{\text{MAP}}$.

middle: Distribution of the marginal preactivations, $p(\mathbf{f}|\mathbf{x})$. The GLM predictive with GGN posterior yields reasonable linearized preactivations $\mathbf{f}_{\text{lin}}^{\boldsymbol{\theta}^*}(\mathbf{x}, \boldsymbol{\theta})$, while the other Laplace distributions already give rise to broader distributions because they use the original $\mathbf{f}(\mathbf{x}, \boldsymbol{\theta})$ combined with posteriors that extend to $\mathbf{w} < 0$. ▮ innermost 50%/66% of samples; — median.

bottom: Posterior predictives $p(\mathbf{y}|\mathbf{f}(\mathbf{x}))$. The GLM predictive with Laplace-GGN posterior closely resembles the true (HMC) predictive; the BNN predictive with any of the Laplace posteriors underfit. ▮ innermost 50%/66% of samples; — median, --- mean.

B.2 Illustrative $2d$ example: the banana dataset

In Sec. 4.1, we considered binary classification on the synthetic banana dataset. Here we provide further experimental details and results.

B.2.1 Experimental details

We use a synthetic dataset known as 'banana' and separate 5% ($N = 265$) of it as training data and 5% as validation set. 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 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.

B.2.2 Aleatoric and epistemic uncertainty

We can decompose the overall uncertainty into aleatoric and epistemic uncertainty. 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 (e.g. missing data) and can be minimized by sampling more data. Therefore, the decomposition into aleatoric and epistemic uncertainty allows us to establish to what extent a model is uncertain about its predictions is due to inherent noise in the data and to what extend the uncertainty is due to the lack of data. This 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), who derive the following decomposition of the total variance into aleatoric and epistemic uncertainty:

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

$$(\text{B.1})$$

B.2.3 Additional experimental results

Here, we present additional results using different posterior approximations (diagonal and full covariance Laplace-GGN approximation and MFVI as well as MAP) and predictives (GLM predictive, BNN predictive) as well as on different architectures.

In Fig. B.3 we show the GLM with Laplace-GGN posterior or a refined GGN posterior with variational inference. In addition to the full covariance Laplace-GGN, we also consider diagonal posterior approximations. In general, the Laplace approximation results in less confident predictions than the refined posterior, which is trained using variational inference. We attribute this to the mode seeking behaviour of variational inference that likely results in a narrower posterior.

We also find that the diagonal posterior approximations generally performs worse compared to the full covariance posteriors; for the Laplace the diagonal posterior is underconfident, whereas for the refined posterior it is overconfident. In both cases the epistemic and aleatoric uncertainties are not very meaningful for the diagonal posterior. We attribute this to the diagonal approximation neglecting important correlations between some of the weights, which are captured by the full covariance posteriors. This highlights that diagonal approximations are often too crude to be useful in practise. While the two diagonal posteriors behave very differently, we find that the full covariance versions behave relatively similar.

In Figs. B.4 and B.5, we compare the proposed GLM predictive with Laplace-GGN posterior to the BNN predictive with same posterior as well as to the NN MAP and mean field variational (BBB) for networks with one and three layers, respectively.
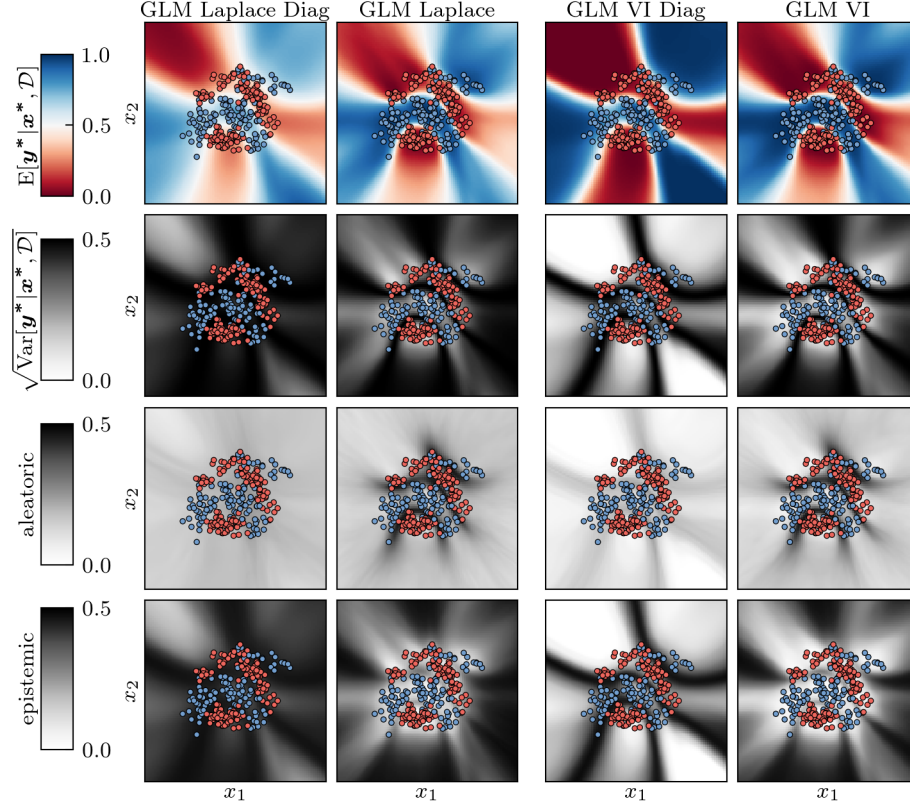
Figure B.3: *From top to bottom:* Mean, standard deviation, aleatoric uncertainty, and epistemic uncertainty on the banana dataset for the GLM predictive. We compare full and diagonal covariances for the Laplace-GGN posterior and a posterior with additional VI refinement for a network with **2 hidden layers with 50 units and `tanh` activation function**.

A lower number of parameters (1 layer, Fig. B.4) slightly improves the performance of BNN predictive, but still the variance is severely overestimated. The GLM predictive has higher variance compared to the model with two layers (Fig. 4) similarly to NN MAP on which it was based. Mean field VI behaves reasonably, though it tends to be overly confident away from the data.

When the model is greatly overparameterized (3 layers, Fig. B.5), the BNN predictive completely fails even for the diagonal approximation. In contrast, the GLM predictive becomes somewhat more confident (similarly to the NN MAP), though it still maintains reasonable uncertainties away from the data. Mean field VI was very hard to train in this overparameterized setting; we had to reduce the layer width dramatically in order for it not to underfit, and show results for a much narrower network, which is nonetheless overly confident. We attribute this behaviour of MFVI to the large KL penalty in the ELBO objective, which becomes more egregious the larger the number of parameters becomes relative to the number of datapoints.
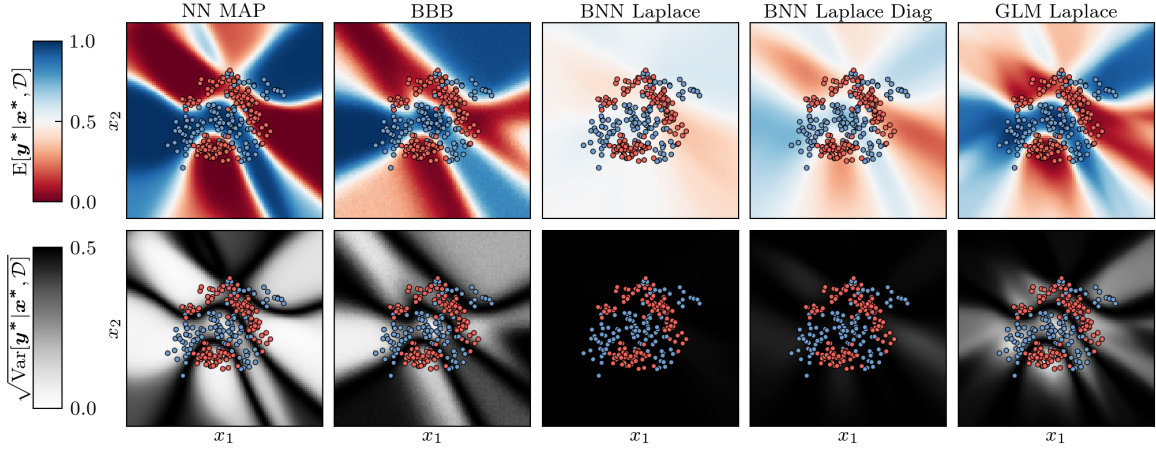
Figure B.4: Mean and standard deviation of the predictives on the banana dataset for a network with **1 hidden layer with 50 units and `tanh` activation** function. A significant decrease of parameters compared to a network with 2 layers results in a decrease in confidence of NN MAP as well as the corresponding GLM predictions; the BNN Laplace predictive improves slightly, but the variance is still greatly overestimated.
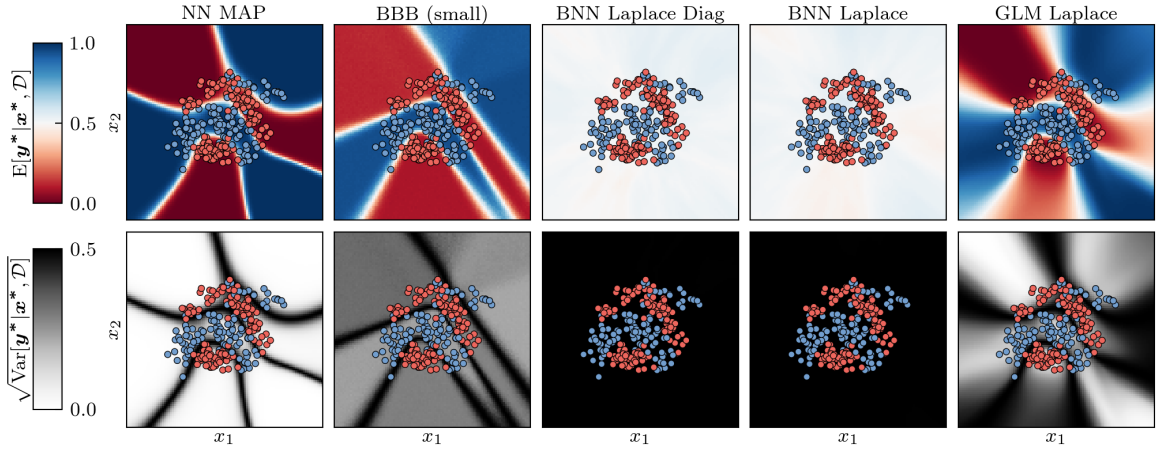
Figure B.5: Mean and standard deviation of the predictives on the banana dataset for a network with **3 hidden layers with 50 units each and `tanh` activation function** (with the exception of BBB where number of hidden units per layer was limited to 5 due to failing to fit the data with more units per layer). Increasing the model capacity leads to more confident predictions for the NN MAP and the corresponding GLM predictive (narrower decision boundary), yet the GLM predictive is still able to produce reasonable uncertainties away from data.

B.3   Classification on UCI

In Sec. 4.2 we considered a suite of binary and multi-class classification tasks; here we provide details about the datasets, further experimental details, as well as additional experimental results and different evaluation metrics.

We use the following 8 UCI classification data sets available from UCI Machine Learning Repository: `https://archive.ics.uci.edu/ml/datasets.php` for our comparison, see Tab. B1

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

B.3.1   Experimental details

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 and Hagan, 1997) (Eq. 9) with our GLM and GP predictive compared to the BNN predictive. We compare both a refined Laplace and Gaussian variational approximation of the GLM formulation (Sec. 3) with 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 and Ba, 2015) using a learning rate of 10^{-3} for MAP and 10^{-3} for BBB.

For refinement in the GLM, we use 1000 iterations for the Laplace approximation and 250 for the variational approximations. For the Laplace approximation, we 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 two 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.

B.3.2   Additional results

Tab. B2 complements Tab. 1 with results on accuracy and expected calibration error (ECE) which measures how well predicted uncertainty corresponds to empirical accuracy (Naeini et al., 2015). We use 10 bins to estimate the ECE. The other metrics reveal that the MAP provides good performance on accuracy despite being worse than the GLM predictive on NLL; the GLM predictive is best calibrated and overall outperforms the BNN predictive as expected.

In Tab. B3, we additionally show results on a shallower, single hidden layer model which is expected to be beneficial for MFVI since it is brittle and hard to tune. With a single hidden layer, the performance of the all methods generally goes down slightly while the performance of MFVI goes up. Nonetheless, the GLM predictive and MAP still perform best and are preferred to MFVI due to lower runtime but overall better performance.

Dataset	NN MAP	MFVI	BNN	GLM	GLM diag	GLM refine	GLM refine d
australian	$\mathbf{0.31}_{\pm\mathbf{0.01}}$	$0.34_{\pm0.01}$	$0.42_{\pm0.00}$	$\mathbf{0.32}_{\pm\mathbf{0.02}}$	$0.33_{\pm0.01}$	$\mathbf{0.32}_{\pm\mathbf{0.02}}$	$\mathbf{0.31}_{\pm\mathbf{0.01}}$
cancer	$\mathbf{0.11}_{\pm\mathbf{0.02}}$	$\mathbf{0.11}_{\pm\mathbf{0.01}}$	$0.19_{\pm0.00}$	$\mathbf{0.10}_{\pm\mathbf{0.01}}$	$\mathbf{0.11}_{\pm\mathbf{0.01}}$	$\mathbf{0.11}_{\pm\mathbf{0.01}}$	$0.12_{\pm0.02}$
ionosphere	$0.35_{\pm0.02}$	$0.41_{\pm0.01}$	$0.50_{\pm0.00}$	$\mathbf{0.29}_{\pm\mathbf{0.01}}$	$0.35_{\pm0.01}$	$0.35_{\pm0.05}$	$0.32_{\pm0.03}$
glass	$0.95_{\pm0.03}$	$1.06_{\pm0.01}$	$1.41_{\pm0.00}$	$0.86_{\pm0.01}$	$0.99_{\pm0.01}$	$0.98_{\pm0.07}$	$\mathbf{0.83}_{\pm\mathbf{0.02}}$
vehicle	$0.420_{\pm0.007}$	$0.504_{\pm0.006}$	$0.885_{\pm0.002}$	$0.428_{\pm0.005}$	$0.618_{\pm0.003}$	$\mathbf{0.402}_{\pm\mathbf{0.007}}$	$0.432_{\pm0.005}$
waveform	$\mathbf{0.335}_{\pm\mathbf{0.004}}$	$0.393_{\pm0.003}$	$0.516_{\pm0.002}$	$\mathbf{0.339}_{\pm\mathbf{0.004}}$	$0.388_{\pm0.003}$	$\mathbf{0.335}_{\pm\mathbf{0.004}}$	$0.364_{\pm0.008}$
digits	$\mathbf{0.094}_{\pm\mathbf{0.003}}$	$0.219_{\pm0.004}$	$0.875_{\pm0.002}$	$0.250_{\pm0.002}$	$0.409_{\pm0.002}$	$0.150_{\pm0.002}$	$0.149_{\pm0.008}$
satellite	$0.230_{\pm0.002}$	$0.307_{\pm0.002}$	$0.482_{\pm0.001}$	$0.241_{\pm0.001}$	$0.327_{\pm0.002}$	$\mathbf{0.227}_{\pm\mathbf{0.002}}$	$0.248_{\pm0.002}$

(a) Negative test log likelihood

Dataset	NN MAP	MFVI	BNN	GLM	GLM diag	GLM refine	GLM ref. d
australian	$0.884_{\pm0.010}$	$0.885_{\pm0.008}$	$0.887_{\pm0.009}$	$0.883_{\pm0.009}$	$0.885_{\pm0.009}$	$\mathbf{0.888}_{\pm\mathbf{0.008}}$	$0.886_{\pm0.008}$
cancer	$0.971_{\pm0.004}$	$0.969_{\pm0.003}$	$\mathbf{0.972}_{\pm\mathbf{0.003}}$	$0.969_{\pm0.003}$	$0.969_{\pm0.003}$	$0.971_{\pm0.003}$	$0.971_{\pm0.003}$
ionosphere	$0.887_{\pm0.006}$	$0.879_{\pm0.007}$	$0.866_{\pm0.009}$	$0.887_{\pm0.009}$	$0.883_{\pm0.008}$	$0.891_{\pm0.007}$	$\mathbf{0.892}_{\pm\mathbf{0.005}}$
glass	$\mathbf{0.684}_{\pm\mathbf{0.010}}$	$0.534_{\pm0.013}$	$0.459_{\pm0.009}$	$0.678_{\pm0.013}$	$0.666_{\pm0.013}$	$0.675_{\pm0.015}$	$0.669_{\pm0.012}$
vehicle	$0.827_{\pm0.005}$	$0.717_{\pm0.003}$	$0.712_{\pm0.003}$	$\mathbf{0.828}_{\pm\mathbf{0.005}}$	$0.814_{\pm0.006}$	$0.824_{\pm0.005}$	$0.820_{\pm0.005}$
waveform	$0.855_{\pm0.003}$	$\mathbf{0.861}_{\pm\mathbf{0.003}}$	$0.858_{\pm0.003}$	$0.854_{\pm0.003}$	$0.852_{\pm0.004}$	$0.853_{\pm0.003}$	$0.849_{\pm0.003}$
digits	$\mathbf{0.974}_{\pm\mathbf{0.001}}$	$0.951_{\pm0.002}$	$0.924_{\pm0.001}$	$0.964_{\pm0.001}$	$0.960_{\pm0.001}$	$0.966_{\pm0.001}$	$0.970_{\pm0.001}$
satellite	$\mathbf{0.916}_{\pm\mathbf{0.001}}$	$0.891_{\pm0.001}$	$0.842_{\pm0.001}$	$0.915_{\pm0.001}$	$0.910_{\pm0.001}$	$\mathbf{0.917}_{\pm\mathbf{0.001}}$	$0.911_{\pm0.001}$

(b) Test accuracy

Dataset	NN MAP	MFVI	BNN	GLM	GLM diag	GLM refine	GLM ref. d
australian	$0.056_{\pm0.004}$	$0.099_{\pm0.005}$	$0.185_{\pm0.008}$	$0.055_{\pm0.005}$	$0.094_{\pm0.007}$	$0.060_{\pm0.005}$	$\mathbf{0.050}_{\pm\mathbf{0.003}}$
cancer	$0.033_{\pm0.002}$	$0.033_{\pm0.002}$	$0.127_{\pm0.003}$	$0.033_{\pm0.002}$	$0.053_{\pm0.003}$	$\mathbf{0.027}_{\pm\mathbf{0.003}}$	$\mathbf{0.027}_{\pm\mathbf{0.003}}$
ionosphere	$0.092_{\pm0.004}$	$0.155_{\pm0.009}$	$0.238_{\pm0.008}$	$0.089_{\pm0.003}$	$0.131_{\pm0.006}$	$\mathbf{0.081}_{\pm\mathbf{0.004}}$	$\mathbf{0.081}_{\pm\mathbf{0.004}}$
glass	$0.192_{\pm0.008}$	$\mathbf{0.140}_{\pm\mathbf{0.006}}$	$0.175_{\pm0.009}$	$0.159_{\pm0.007}$	$0.222_{\pm0.012}$	$0.170_{\pm0.007}$	$0.160_{\pm0.005}$
vehicle	$0.064_{\pm0.003}$	$0.058_{\pm0.002}$	$0.266_{\pm0.003}$	$0.078_{\pm0.003}$	$0.210_{\pm0.005}$	$\mathbf{0.052}_{\pm\mathbf{0.002}}$	$0.071_{\pm0.002}$
waveform	$0.041_{\pm0.002}$	$0.134_{\pm0.002}$	$0.233_{\pm0.003}$	$0.049_{\pm0.002}$	$0.109_{\pm0.004}$	$\mathbf{0.038}_{\pm\mathbf{0.001}}$	$0.049_{\pm0.002}$
digits	$\mathbf{0.016}_{\pm\mathbf{0.001}}$	$0.091_{\pm0.002}$	$0.478_{\pm0.001}$	$0.141_{\pm0.000}$	$0.256_{\pm0.001}$	$0.054_{\pm0.001}$	$0.027_{\pm0.000}$
satellite	$\mathbf{0.018}_{\pm\mathbf{0.001}}$	$0.033_{\pm0.001}$	$0.125_{\pm0.001}$	$0.037_{\pm0.001}$	$0.118_{\pm0.001}$	$\mathbf{0.019}_{\pm\mathbf{0.001}}$	$0.022_{\pm0.001}$

(c) Test expected calibration error

Table B2: Additional results on the UCI classification experiment presented in Sec. 4.2: We have a **2 hidden layer MLP with `tanh` activation function and 50 neurons each**. In Table (b), we additionally show results on accuracy and in (c) on expected calibration error (ECE). The MAP estimate works best on most datasets in terms of accuracy but the GLM predictives are best calibrated and attain a better log likelihood.

Dataset	NN MAP	MFVI	BNN	GLM	GLM diag	GLM refine	GLM ref. d
australian	$\mathbf{0.32}_{\pm\mathbf{0.01}}$	$\mathbf{0.32}_{\pm\mathbf{0.01}}$	$0.38_{\pm0.00}$	$\mathbf{0.32}_{\pm\mathbf{0.01}}$	$0.33_{\pm0.01}$	$\mathbf{0.32}_{\pm\mathbf{0.01}}$	$\mathbf{0.31}_{\pm\mathbf{0.01}}$
cancer	$0.15_{\pm0.03}$	$\mathbf{0.11}_{\pm\mathbf{0.01}}$	$0.15_{\pm0.00}$	$\mathbf{0.10}_{\pm\mathbf{0.01}}$	$\mathbf{0.11}_{\pm\mathbf{0.01}}$	$0.17_{\pm0.05}$	$0.16_{\pm0.05}$
ionosphere	$0.32_{\pm0.02}$	$\mathbf{0.27}_{\pm\mathbf{0.02}}$	$0.45_{\pm0.00}$	$0.29_{\pm0.01}$	$0.35_{\pm0.01}$	$\mathbf{0.27}_{\pm\mathbf{0.02}}$	$\mathbf{0.26}_{\pm\mathbf{0.02}}$
glass	$0.91_{\pm0.03}$	$0.90_{\pm0.06}$	$1.25_{\pm0.00}$	$0.87_{\pm0.01}$	$0.96_{\pm0.01}$	$\mathbf{0.75}_{\pm\mathbf{0.02}}$	$0.79_{\pm0.03}$
vehicle	$0.433_{\pm0.007}$	$\mathbf{0.394}_{\pm\mathbf{0.005}}$	$0.795_{\pm0.003}$	$0.451_{\pm0.004}$	$0.587_{\pm0.004}$	$\mathbf{0.396}_{\pm\mathbf{0.007}}$	$0.421_{\pm0.006}$
waveform	$\mathbf{0.338}_{\pm\mathbf{0.004}}$	$0.360_{\pm0.007}$	$0.434_{\pm0.002}$	$\mathbf{0.342}_{\pm\mathbf{0.004}}$	$0.368_{\pm0.003}$	$\mathbf{0.339}_{\pm\mathbf{0.004}}$	$0.343_{\pm0.005}$
digits	$\mathbf{0.086}_{\pm\mathbf{0.003}}$	$0.137_{\pm0.004}$	$0.671_{\pm0.002}$	$0.256_{\pm0.002}$	$0.401_{\pm0.003}$	$0.170_{\pm0.011}$	$0.143_{\pm0.004}$
satellite	$0.222_{\pm0.002}$	$0.274_{\pm0.002}$	$0.429_{\pm0.001}$	$0.240_{\pm0.001}$	$0.281_{\pm0.001}$	$\mathbf{0.219}_{\pm\mathbf{0.001}}$	$0.238_{\pm0.002}$

(a) Negative test log likelihood

Dataset	NN MAP	MFVI	BNN	GLM	GLM diag	GLM refine	GLM ref. d
australian	$\mathbf{0.881}_{\pm\mathbf{0.010}}$	$\mathbf{0.883}_{\pm\mathbf{0.008}}$	$\mathbf{0.888}_{\pm\mathbf{0.008}}$	$\mathbf{0.882}_{\pm\mathbf{0.010}}$	$0.879_{\pm0.010}$	$\mathbf{0.880}_{\pm\mathbf{0.010}}$	$\mathbf{0.880}_{\pm\mathbf{0.009}}$
cancer	$0.967_{\pm0.004}$	$\mathbf{0.970}_{\pm\mathbf{0.003}}$	$\mathbf{0.973}_{\pm\mathbf{0.003}}$	$0.968_{\pm0.003}$	$\mathbf{0.972}_{\pm\mathbf{0.003}}$	$0.963_{\pm0.003}$	$\mathbf{0.971}_{\pm\mathbf{0.003}}$
ionosphere	$0.875_{\pm0.010}$	$\mathbf{0.909}_{\pm\mathbf{0.007}}$	$0.881_{\pm0.008}$	$0.872_{\pm0.010}$	$0.883_{\pm0.009}$	$0.898_{\pm0.007}$	$0.896_{\pm0.007}$
glass	$\mathbf{0.706}_{\pm\mathbf{0.011}}$	$0.672_{\pm0.015}$	$0.550_{\pm0.014}$	$0.691_{\pm0.010}$	$0.691_{\pm0.013}$	$\mathbf{0.700}_{\pm\mathbf{0.013}}$	$0.669_{\pm0.013}$
vehicle	$0.809_{\pm0.005}$	$0.802_{\pm0.003}$	$0.750_{\pm0.003}$	$0.815_{\pm0.003}$	$0.817_{\pm0.004}$	$\mathbf{0.827}_{\pm\mathbf{0.004}}$	$0.808_{\pm0.004}$
waveform	$\mathbf{0.862}_{\pm\mathbf{0.003}}$	$0.852_{\pm0.003}$	$0.858_{\pm0.002}$	$\mathbf{0.860}_{\pm\mathbf{0.003}}$	$0.845_{\pm0.003}$	$0.857_{\pm0.003}$	$\mathbf{0.859}_{\pm\mathbf{0.003}}$
digits	$\mathbf{0.977}_{\pm\mathbf{0.001}}$	$0.969_{\pm0.001}$	$0.954_{\pm0.001}$	$0.971_{\pm0.001}$	$0.963_{\pm0.001}$	$0.966_{\pm0.001}$	$0.967_{\pm0.001}$
satellite	$\mathbf{0.919}_{\pm\mathbf{0.001}}$	$0.900_{\pm0.001}$	$0.863_{\pm0.001}$	$\mathbf{0.918}_{\pm\mathbf{0.001}}$	$0.917_{\pm0.001}$	$\mathbf{0.918}_{\pm\mathbf{0.001}}$	$0.912_{\pm0.001}$

(b) Test accuracy

Dataset	NN MAP	MFVI	BNN	GLM	GLM diag	GLM refine	GLM ref. d
australian	$0.063_{\pm0.005}$	$0.061_{\pm0.003}$	$0.144_{\pm0.007}$	$0.061_{\pm0.006}$	$0.082_{\pm0.006}$	$\mathbf{0.055}_{\pm\mathbf{0.005}}$	$\mathbf{0.058}_{\pm\mathbf{0.003}}$
cancer	$0.035_{\pm0.003}$	$0.032_{\pm0.003}$	$0.086_{\pm0.004}$	$0.034_{\pm0.003}$	$0.048_{\pm0.003}$	$\mathbf{0.026}_{\pm\mathbf{0.002}}$	$\mathbf{0.027}_{\pm\mathbf{0.002}}$
ionosphere	$0.080_{\pm0.006}$	$0.083_{\pm0.002}$	$0.214_{\pm0.006}$	$0.081_{\pm0.004}$	$0.132_{\pm0.007}$	$0.077_{\pm0.004}$	$\mathbf{0.067}_{\pm\mathbf{0.005}}$
glass	$\mathbf{0.155}_{\pm\mathbf{0.010}}$	$0.180_{\pm0.008}$	$0.215_{\pm0.010}$	$0.173_{\pm0.008}$	$0.232_{\pm0.010}$	$\mathbf{0.157}_{\pm\mathbf{0.005}}$	$0.189_{\pm0.008}$
vehicle	$0.078_{\pm0.005}$	$\mathbf{0.060}_{\pm\mathbf{0.002}}$	$0.255_{\pm0.003}$	$0.088_{\pm0.003}$	$0.181_{\pm0.005}$	$0.065_{\pm0.003}$	$0.070_{\pm0.003}$
waveform	$\mathbf{0.044}_{\pm\mathbf{0.002}}$	$0.064_{\pm0.002}$	$0.160_{\pm0.002}$	$0.055_{\pm0.001}$	$0.078_{\pm0.003}$	$0.052_{\pm0.002}$	$0.049_{\pm0.002}$
digits	$\mathbf{0.014}_{\pm\mathbf{0.000}}$	$0.041_{\pm0.001}$	$0.410_{\pm0.001}$	$0.156_{\pm0.001}$	$0.257_{\pm0.001}$	$0.049_{\pm0.002}$	$0.033_{\pm0.002}$
satellite	$0.020_{\pm0.000}$	$0.021_{\pm0.001}$	$0.119_{\pm0.001}$	$0.044_{\pm0.001}$	$0.090_{\pm0.001}$	$\mathbf{0.018}_{\pm\mathbf{0.001}}$	$0.021_{\pm0.001}$

(c) Test expected calibration error

Table B3: Performance on UCI classification task with **a smaller single hidden layer and `tanh` activation**. The GLM predictives remain consistently better than the BNN predictives. The GLM predictives are also preferred to MFVI, even when only considering a diagonal refined posterior which is also cheaper than MFVI.

B.4 Image classification

For the large scale image benchmarks, we use several architectures, hyperparameters, and methods. Here, we show and discuss additional results to the ones presented in Sec. 4.3 and Tab. 2. After, we describe the used architectures that are similar to the ones in the DeepOBS benchmark suite (Schneider et al., 2018). We evaluate and compare our methods on the three common image benchmarks MNIST (LeCun and Cortes, 2010), FashionMNIST (Xiao et al., 2017), and CIFAR-10 (Krizhevsky, 2009).

B.4.1 Additional results

In Tab. B4, additional performance results are listed that are in line with the results presented in Sec. 4.3: the GLM predictive is preferred across tasks, datasets, architectures, and performance metrics overall. It provides significantly better results than the BNN predictive as expected: across all tasks, we find that the BNN predictive without posterior concentration underfits extremely due to the mismatched predictive model. Both, the proposed GLM and GP predictive fix this problem. The GLM predictive typically performs best in terms of accuracy and OOD detection while the GP predictive gives strong results on NLL and is very well calibrated. Dampening the posterior as done by Ritter et al. (2018) improves the performance of the BNN predictive in some cases but not consistently; for example, using an MLP yields bad performance despite dampening. We additionally present results on a simple diagonal posterior approximation. Here, the GLM predictive also works consistently better than the BNN predictive.

B.4.2 Architectures

We use different architectures depending on the data sets: For MNIST and FMNIST, we use one fully connected and one mixed architecture (convolutional and fully connected layers). On CIFAR-10, we use the same mixed architecture and a fully convolutional architecture.

For the MNIST and FMNIST data sets, we train a multilayer perceptron (MLP) with 4 hidden layers of sizes $[1024, 512, 256, 128]$. The MLP is an interesting benchmark due to the number of parameters per layer and the potential problem of distorted predictives.

On both data sets, we also use a standard mixed architecture with three convolutional and three fully connected layers as implemented in the DeepOBS benchmark suite (Schneider et al., 2018).

On CIFAR-10, we do not use an MLP as it is hard to fit and performs significantly worse than convolutional architectures. Instead, we use the same mixed architecture (CNN) and another fully convolutional architecture (AllCNN), which is again as standardized by Schneider et al. (2018). The AllCNN architecture uses 9 convolutional blocks followed by average pooling. In contrast to Schneider et al. (2018), we do not use additional dropout. All convolutional layers are followed by `ReLU` activation and fully connected layers by `tanh` activation, except for the final layers.

B.4.3 MAP estimation

We train all our models in `pytorch` (Paszke et al., 2017) using the Adam optimizer (Kingma and Ba, 2015) on the log joint objective in Eq. (3). We use the default Adam learning rate of 10^{-3} and run the training procedure for 500 epochs to ensure convergence and use a batch size of 512. In line with standard Bayesian deep learning methods, we assume an isotropic Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(0, \delta^{-1}\boldsymbol{I})$ on all parameters. To estimate standard errors presented in the results, we run every experiment with 5 different random seeds. Except in the CIFAR-10 experiment with the fully convolutional architecture which is expensive to train, we train each for 16 values of δ log-spaced between 10^{-2} and 10^3. On the fully convolutional network, we train 6 networks.

B.4.4 Inference

We construct the considered Laplace-GGN posterior approximations on all the trained models and evaluate their performance on a validation set to select the best prior precision δ per model-dataset-method combination. The KFAC and DIAG approximations to the GGN are computed using `backpack` for `pytorch` (Dangel et al., 2019). The GP subset of data Laplace approximation is constructed only on the best MAP network per seed and we compare the impact of different subset sizes M below.

B.4.5   Ablation study: Subset-of-data GP approximation

As described in App. A.2, we use a subset of data GP predictive with M training samples to construct the posterior and the posterior predictive. Here, we investigate the influence of this subset size on performance, see Fig. B.6 for a comparison against the neural network MAP. As already observed in the experimental results, the GP predictive greatly improves the calibration in terms of ECE but also improves the NLL over the MAP baseline network even with as few as $M = 50$ data points. Further, the improvement typically consistently increases with improved subset size M. We consider two strategies for selecting the subset of data. We either sample M training data points at random or use the top-M selection criterion as proposed by Pan et al. (2020) for a GP arising from DNN2GP. In line with Pan et al. (2020), we find that selecting the top-M relevant points is mostly beneficial for small M. However, we also find that selecting M points uniformly at random is sufficient for good performance and is slightly cheaper as we do not need to pass through the entire training data.
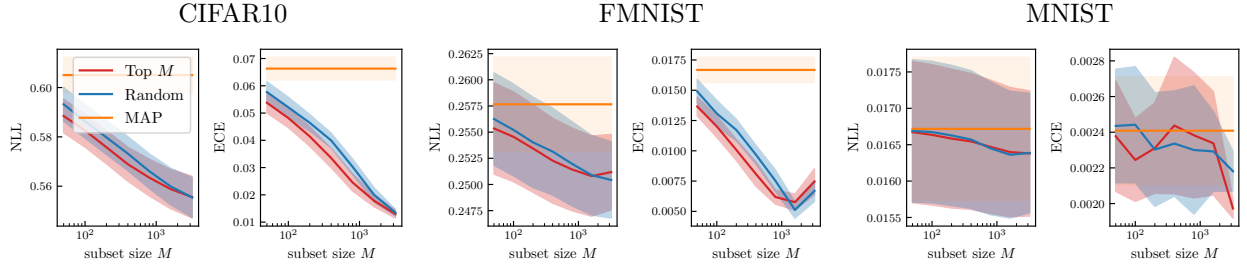

Figure B.6: Performance vs. subset size M for the GP predictive. We compare the GP predictive to the respective MAP baseline in terms of negative log likelihood (NLL) and expected calibration error (ECE) and compare a *random* (⬛) to the *top-M* (⬛, (Pan et al., 2020)) subset selection approach. Both selection methods lead to performance improvements where possible, especially with larger subset sizes M. On CIFAR10 and FMNIST we can obtain considerable improvements over the MAP, especially in terms of calibration where the improvement is up to 6-fold. On MNIST, the MAP is already sufficient and there is no room for improvement. We show results on CIFAR10 (*left*), FMNIST (*middle*), and MNIST (*right*) using the best performing architectures on the respective tasks (see Tab. B4).

B.4.6   Ablation study: Predictive distribution for dampened posteriors (Ritter et al., 2018)

Finally, we investigate how well the dampened KFAC Laplace-GGN posterior by Ritter et al. (2018) performs when used in conjunction with our proposed GLM predictive instead of the BNN predictive originally used by the authors, see Tab. B5 for results. The performance is not consistent over all metrics; on accuracy, the GLM predictive always improves over the BNN predictive, especially on fully connected networks.

Note that dampening corresponds to an ad-hoc approximation that is not necessary in this setting; using the un-dampened posterior with GLM predictive generally performs similar or better on most datasets.

Dataset	Model	Method	Accuracy ↑	NLL ↓	ECE ↓	OD-AUC ↑
MNIST	**MLP**	MAP	$\mathbf{98.22}_{\pm0.13}$	$0.061_{\pm0.004}$	$0.006_{\pm0.001}$	$0.806_{\pm0.015}$
		BNN predictive DIAG	$93.03_{\pm0.13}$	$0.369_{\pm0.003}$	$0.168_{\pm0.001}$	$0.902_{\pm0.002}$
		BNN predictive KFAC	$92.89_{\pm0.12}$	$0.576_{\pm0.008}$	$0.315_{\pm0.006}$	$0.885_{\pm0.004}$
		BNN predictive (Ritter et al.)	$93.14_{\pm0.05}$	$0.304_{\pm0.002}$	$0.111_{\pm0.003}$	$0.927_{\pm0.001}$
		GLM predictive DIAG (*ours*)	$96.58_{\pm0.20}$	$0.189_{\pm0.002}$	$0.094_{\pm0.005}$	$0.913_{\pm0.003}$
		GLM predictive KFAC (*ours*)	$\mathbf{98.40}_{\pm0.05}$	$\mathbf{0.054}_{\pm0.002}$	$0.007_{\pm0.001}$	$\mathbf{0.970}_{\pm0.002}$
		GP predictive (*ours*)	$\mathbf{98.22}_{\pm0.13}$	$0.058_{\pm0.001}$	$\mathbf{0.003}_{\pm0.000}$	$0.809_{\pm0.014}$
	CNN	MAP	$\mathbf{99.40}_{\pm0.03}$	$\mathbf{0.017}_{\pm0.001}$	$\mathbf{0.002}_{\pm0.000}$	$0.989_{\pm0.00}$
		BNN predictive DIAG	$97.78_{\pm0.03}$	$0.947_{\pm0.005}$	$0.576_{\pm0.002}$	$0.939_{\pm0.004}$
		BNN predictive KFAC	$97.75_{\pm0.08}$	$1.176_{\pm0.026}$	$0.660_{\pm0.007}$	$0.938_{\pm0.003}$
		BNN predictive (Ritter et al.)	$99.06_{\pm0.04}$	$0.055_{\pm0.001}$	$0.031_{\pm0.001}$	$\mathbf{0.993}_{\pm0.001}$
		GLM predictive DIAG (*ours*)	$98.12_{\pm0.05}$	$0.160_{\pm0.002}$	$0.103_{\pm0.002}$	$0.981_{\pm0.001}$
		GLM predictive KFAC (*ours*)	$\mathbf{99.40}_{\pm0.02}$	$\mathbf{0.017}_{\pm0.001}$	$\mathbf{0.003}_{\pm0.000}$	$0.990_{\pm0.001}$
		GP predictive (*ours*)	$\mathbf{99.41}_{\pm0.03}$	$\mathbf{0.016}_{\pm0.001}$	$\mathbf{0.002}_{\pm0.000}$	$0.989_{\pm0.001}$
FMNIST	**MLP**	MAP	$88.09_{\pm0.10}$	$0.347_{\pm0.005}$	$0.026_{\pm0.004}$	$0.869_{\pm0.006}$
		BNN predictive DIAG	$83.08_{\pm0.14}$	$0.602_{\pm0.002}$	$0.170_{\pm0.002}$	$0.906_{\pm0.009}$
		BNN predictive KFAC	$81.98_{\pm0.28}$	$0.842_{\pm0.009}$	$0.319_{\pm0.004}$	$0.919_{\pm0.009}$
		BNN predictive (Ritter et al.)	$83.34_{\pm0.13}$	$0.518_{\pm0.003}$	$0.094_{\pm0.002}$	$\mathbf{0.949}_{\pm0.006}$
		GLM predictive DIAG (*ours*)	$87.50_{\pm0.22}$	$0.418_{\pm0.004}$	$0.110_{\pm0.002}$	$0.910_{\pm0.003}$
		GLM predictive KFAC (*ours*)	$\mathbf{88.64}_{\pm0.27}$	$\mathbf{0.340}_{\pm0.006}$	$0.021_{\pm0.004}$	$0.907_{\pm0.020}$
		GP predictive (*ours*)	$88.04_{\pm0.09}$	$\mathbf{0.338}_{\pm0.003}$	$\mathbf{0.010}_{\pm0.001}$	$0.889_{\pm0.006}$
	CNN	MAP	$91.39_{\pm0.11}$	$0.258_{\pm0.004}$	$0.017_{\pm0.001}$	$0.864_{\pm0.014}$
		BNN predictive DIAG	$84.37_{\pm0.11}$	$0.809_{\pm0.002}$	$0.340_{\pm0.002}$	$0.924_{\pm0.007}$
		BNN predictive KFAC	$84.42_{\pm0.12}$	$0.942_{\pm0.016}$	$0.411_{\pm0.008}$	$0.945_{\pm0.002}$
		BNN predictive (Ritter et al.)	$91.20_{\pm0.07}$	$0.265_{\pm0.004}$	$0.024_{\pm0.002}$	$0.947_{\pm0.006}$
		GLM predictive DIAG (*ours*)	$89.45_{\pm0.14}$	$0.397_{\pm0.002}$	$0.136_{\pm0.002}$	$0.944_{\pm0.007}$
		GLM predictive KFAC (*ours*)	$\mathbf{92.25}_{\pm0.10}$	$\mathbf{0.244}_{\pm0.003}$	$0.012_{\pm0.003}$	$\mathbf{0.955}_{\pm0.006}$
		GP predictive (*ours*)	$91.36_{\pm0.11}$	$0.250_{\pm0.004}$	$\mathbf{0.007}_{\pm0.001}$	$0.918_{\pm0.010}$
CIFAR10	**CNN**	MAP	$77.41_{\pm0.06}$	$0.680_{\pm0.004}$	$0.045_{\pm0.004}$	$\mathbf{0.809}_{\pm0.006}$
		BNN predictive DIAG	$69.38_{\pm0.25}$	$1.333_{\pm0.003}$	$0.373_{\pm0.004}$	$0.514_{\pm0.008}$
		BNN predictive KFAC	$72.49_{\pm0.20}$	$1.274_{\pm0.010}$	$0.390_{\pm0.003}$	$0.548_{\pm0.005}$
		BNN predictive (Ritter et al.)	$77.38_{\pm0.06}$	$\mathbf{0.661}_{\pm0.003}$	$\mathbf{0.012}_{\pm0.003}$	$0.796_{\pm0.005}$
		GLM predictive DIAG (*ours*)	$\mathbf{77.47}_{\pm0.11}$	$0.783_{\pm0.001}$	$0.184_{\pm0.003}$	$0.699_{\pm0.009}$
		GLM predictive KFAC (*ours*)	$\mathbf{77.44}_{\pm0.05}$	$0.679_{\pm0.004}$	$0.043_{\pm0.003}$	$\mathbf{0.809}_{\pm0.005}$
		GP predictive (*ours*)	$\mathbf{77.42}_{\pm0.05}$	$\mathbf{0.660}_{\pm0.003}$	$0.013_{\pm0.003}$	$0.798_{\pm0.005}$
	AllCNN	MAP	$80.92_{\pm0.32}$	$0.605_{\pm0.007}$	$0.066_{\pm0.004}$	$0.792_{\pm0.008}$
		BNN predictive DIAG	$21.71_{\pm0.79}$	$2.110_{\pm0.021}$	$0.097_{\pm0.011}$	$0.700_{\pm0.019}$
		BNN predictive KFAC	$21.74_{\pm0.80}$	$2.114_{\pm0.021}$	$0.095_{\pm0.012}$	$0.689_{\pm0.020}$
		BNN predictive (Ritter et al.)	$80.78_{\pm0.36}$	$0.588_{\pm0.005}$	$0.052_{\pm0.005}$	$0.783_{\pm0.007}$
		GLM predictive DIAG (*ours*)	$80.26_{\pm0.30}$	$0.968_{\pm0.006}$	$0.348_{\pm0.005}$	$0.743_{\pm0.022}$
		GLM predictive KFAC (*ours*)	$\mathbf{81.37}_{\pm0.15}$	$0.601_{\pm0.008}$	$0.084_{\pm0.010}$	$\mathbf{0.843}_{\pm0.016}$
		GP predictive (*ours*)	$81.01_{\pm0.32}$	$\mathbf{0.555}_{\pm0.008}$	$\mathbf{0.017}_{\pm0.003}$	$0.820_{\pm0.013}$

Table B4: Accuracy, negative test log likelihood (NLL), and expected calibration error (ECE) on the test set as well as area under the curve for out-of-distribution detection (OD-AUC). Bold numbers indicate best performance on particular dataset-model combination. The proposed methods (GLM and GP) typically take the first and second best spot in terms of accuracy and in most cases perform best in the other measured performances as well. In terms of out-of-distribution detection, the GLM provides overall the best performance.

Dataset	Model	Method	Accuracy ↑	NLL ↓	ECE ↓	OD-AUC ↑
MNIST	MLP	BNN predictive	$93.14_{\pm 0.05}$	$0.304_{\pm 0.002}$	$0.111_{\pm 0.003}$	$\mathbf{0.927}_{\pm \mathbf{0.001}}$
		GLM predictive	$\mathbf{98.39}_{\pm \mathbf{0.05}}$	$\mathbf{0.052}_{\pm \mathbf{0.002}}$	$\mathbf{0.004}_{\pm \mathbf{0.000}}$	$0.921_{\pm 0.010}$
	CNN	BNN predictive	$99.06_{\pm 0.04}$	$0.055_{\pm 0.001}$	$0.031_{\pm 0.001}$	$\mathbf{0.993}_{\pm \mathbf{0.001}}$
		GLM predictive	$\mathbf{99.41}_{\pm \mathbf{0.03}}$	$\mathbf{0.016}_{\pm \mathbf{0.001}}$	$\mathbf{0.002}_{\pm \mathbf{0.000}}$	$0.990_{\pm 0.001}$
FMNIST	MLP	BNN predictive	$83.34_{\pm 0.13}$	$0.518_{\pm 0.003}$	$0.094_{\pm 0.002}$	$\mathbf{0.949}_{\pm \mathbf{0.006}}$
		GLM predictive	$\mathbf{88.77}_{\pm \mathbf{0.26}}$	$\mathbf{0.337}_{\pm \mathbf{0.003}}$	$\mathbf{0.022}_{\pm \mathbf{0.004}}$	$0.922_{\pm 0.020}$
	CNN	BNN predictive	$91.20_{\pm 0.07}$	$0.265_{\pm 0.004}$	$0.024_{\pm 0.002}$	$\mathbf{0.947}_{\pm \mathbf{0.006}}$
		GLM predictive	$\mathbf{92.24}_{\pm \mathbf{0.10}}$	$\mathbf{0.242}_{\pm \mathbf{0.004}}$	$\mathbf{0.016}_{\pm \mathbf{0.001}}$	$0.905_{\pm 0.010}$
CIFAR10	CNN	BNN predictive	$77.38_{\pm 0.06}$	$\mathbf{0.661}_{\pm \mathbf{0.003}}$	$\mathbf{0.012}_{\pm \mathbf{0.003}}$	$0.796_{\pm 0.005}$
		GLM predictive	$\mathbf{77.44}_{\pm \mathbf{0.05}}$	$0.679_{\pm 0.004}$	$0.044_{\pm 0.003}$	$\mathbf{0.810}_{\pm \mathbf{0.005}}$
	AllCNN	BNN predictive	$80.78_{\pm 0.36}$	$\mathbf{0.588}_{\pm \mathbf{0.005}}$	$\mathbf{0.052}_{\pm \mathbf{0.005}}$	$0.783_{\pm 0.007}$
		GLM predictive	$\mathbf{81.23}_{\pm \mathbf{0.14}}$	$0.599_{\pm 0.008}$	$0.077_{\pm 0.008}$	$\mathbf{0.843}_{\pm \mathbf{0.016}}$

Table B5: Performance comparison of BNN and GLM predictive using the dampened posterior proposed by (Ritter et al., 2018). On accuracy, the GLM performs strictly better while on other metrics there is a trade-off: either one of the methods is better on OOD-detection or on NLL and ECE but not on both. We find that the GLM predictive also works with the dampenend posterior and in that case is still most of the times the better alternative.

B.5 Out-of-distribution detection

We evaluate the predictives on out-of-distribution (OOD) detection on the following in-distribution (ID)/OOD pairs: MNIST/FMNIST, FMNIST/MNIST, and CIFAR10/SVHN. Following Osawa et al. (2019), we compare the predictive entropies of the distribution on in-distribution (ID) data vs OOD data. To that end, we compute the predictive entropy of the output probability vector $\mathbf{p}$ that is either estimated by sampling; a point estimate for the MAP case can be computed as $-\sum_{c=1}^{C} p_c \log(p_c)$. The predictive entropy is at its maximum in case of a uniform output distribution. Higher predictive entropies are desired for OOD data and lower entropies for ID data. We then use the entropy as a score to distinguish ID and OOD data (Osawa et al., 2019). Varying the score threshold, we can estimate the area under the receiver-operator characteristic as a measure of performance (OOD-AUC).

Figs. B.7 to B.11 show histograms of ID and OOD predictive entropies as well as the OOD-AUC on the different ID/OOD dataset pairs and different architectures. For CIFAR10, we additionally provide results on a mixed architecture (Fig. B.7). On MNIST and FMNIST, we show results on the MLP and mixed architectures: MLP is used in Fig. B.9 and Fig. B.11; mixed architecture in Fig. B.8 and Fig. B.10.

Overall, the GLM predictive provides the best OOD detection performance followed by the dampened BNN predictive by (Ritter et al., 2018). However as the histograms suggest, their predictive is underconfident which is reflected in the results on calibration (ECE) in Tab. B4. The naive BNN predictive performs worst overall. As for other metrics, the GP predictive consistently improves over the MAP estimate.
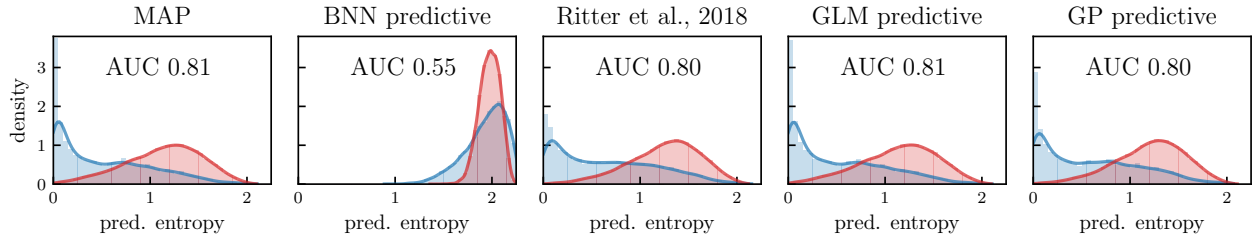

Figure B.7: In-distribution (CIFAR10 ▬) vs out-of-distribution (SVHN ▬) detection and calibration on CIFAR10 using a convolutional architecture with 2 fully connected classification layers. All predictive distributions are well-calibrated and detect OOD data except for the underconfident vanilla BNN predictive.
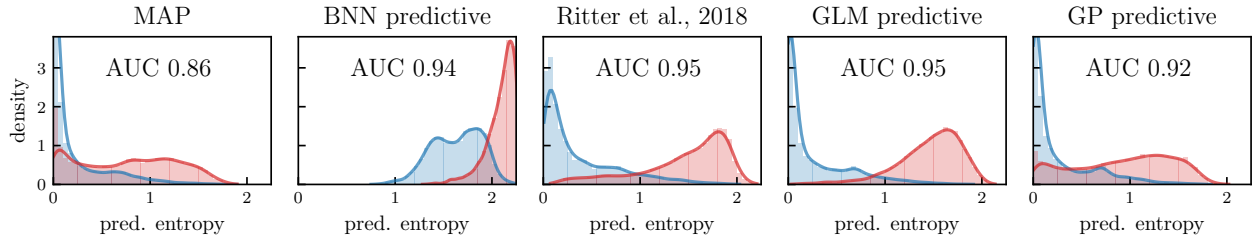

Figure B.8: In-distribution (FMNIST ▬) vs out-of-distribution (MNIST ▬) detection and calibration on FMNIST using convolutional architecture with 2 fully connected classification layers on top. MAP is overconfident while BNN predictive is underconfident but still detects out-of-distribution data well. The GLM predictive performs best.
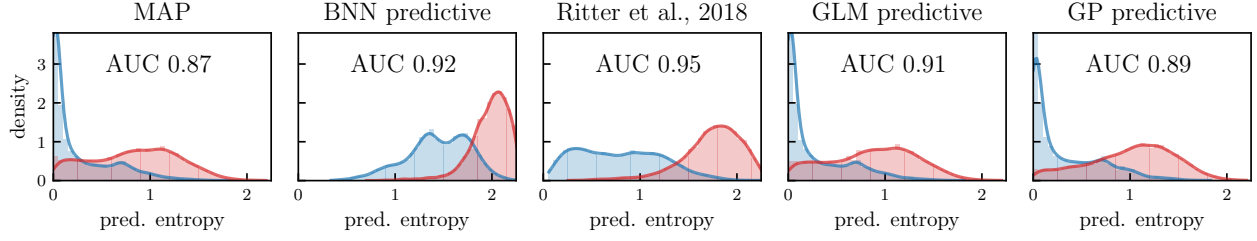
Figure B.9: In-distribution (FMNIST ▬) vs out-of-distribution (MNIST ▬) detection and calibration on FMNIST using fully connected architecture. BNN predictives are underconfident but detect OOD data well. The GP predictive is both well-calibrated and good on OOD detection.
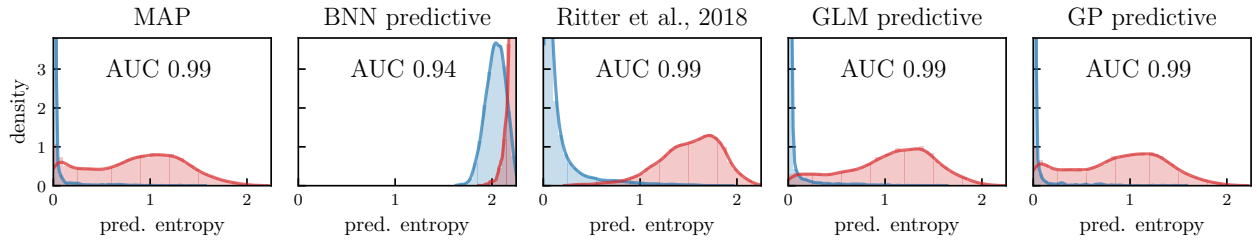

Figure B.10: In-distribution (MNIST ▬) vs out-of-distribution (FMNIST ▬) detection and calibration on MNIST using convolutional architecture with 2 fully connected classification layers. MAP, GLM, and GP predictives are perfecty calibrated and detect OOD data while BNN predictives are slighty suboptimal.
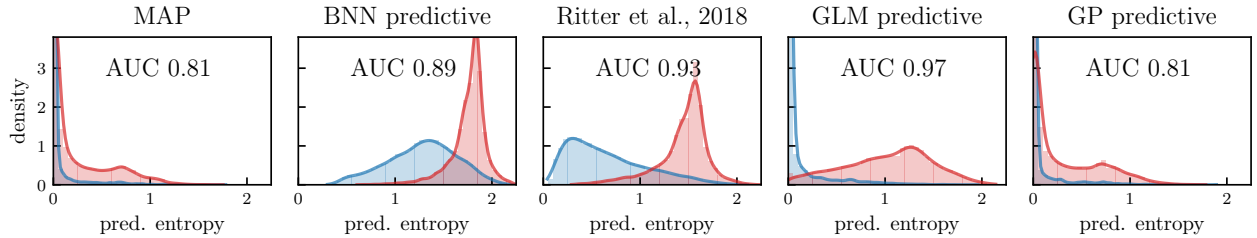

Figure B.11: In-distribution (MNIST ▬) vs out-of-distribution (FMNIST ▬) detection and calibration on MNIST using fully connected architecture. GLM predictive is best on OOD detection and GP is best calibrated. BNN predictives are underconfident while the MAP is overconfident.