An Infinite-Feature Extension for Bayesian ReLU Nets That Fixes Their Asymptotic Overconfidence

Agustinus Kristiadi
University of Tübingen
agustinus.kristiadi@uni-tuebingen.de

Matthias Hein
University of Tübingen
matthias.hein@uni-tuebingen.de

Philipp Hennig
University of Tübingen and MPI for Intelligent Systems, Tübingen
philipp.hennig@uni-tuebingen.de

Abstract

A Bayesian treatment can mitigate overconfidence in ReLU nets around the training data. But far away from them, ReLU Bayesian neural networks (BNNs) can still underestimate uncertainty and thus be asymptotically overconfident. This issue arises since the output variance of a BNN with finitely many features is quadratic in the distance from the data region. Meanwhile, Bayesian linear models with ReLU features converge, in the infinite-width limit, to a particular Gaussian process (GP) with a variance that grows cubically so that no asymptotic overconfidence can occur. While this may seem of mostly theoretical interest, in this work, we show that it can be used concretely to the benefit of BNNs. We extend finite ReLU BNNs with infinite ReLU features via the GP and show that the resulting model is asymptotically maximally uncertain far away from the data while the BNNs' predictive power is unaffected near the data. Although the resulting model approximates a full GP posterior, thanks to its structure it can be applied *post-hoc* to any pre-trained ReLU BNN at a low cost.

1 Introduction

Approximate Bayesian methods, which turn neural nets (NNs) into Bayesian neural networks (BNNs), can be used to mitigate overconfidence. Kristiadi et al. (2020) showed for binary ReLU classification networks that far away from the training data, i.e. when scaling any input with a scalar $\alpha > 0$ and taking the limit $\alpha \to \infty$, the confidence of BNNs can be bounded away from one. This result is encouraging vis-à-vis standard point-estimated networks, for which Hein et al. (2019) showed earlier that the same asymptotic limit always yields arbitrarily high confidence. Nevertheless, BNNs can still be asymptotically overconfident, albeit less so than standard NNs, since the aforementioned uncertainty bound can be loose. Formally, this issue arises because the variance over function outputs of a BNN is asymptotically quadratic w.r.t. α. Since the corresponding mean of ReLU BNNs is asymptotically linear, the growth in uncertainty over their softmax outputs is counteracted by the growing confidence induced by the mean, yielding overconfidence.

Meanwhile there is a particular Gaussian process (GP)—arising from the cubic spline kernel (Wahba, 1990)—which has cubic variance growth and can be seen as a Bayesian linear model with countably infinite ReLU features. We can thus hypothesize that finite ReLU BNNs "miss out" some uncertainty because of their finiteness. In this work, we "add back" this missing uncertainty into finite ReLU BNNs by first extending the cubic spline kernel to cover the whole input domain (Figure 1) and then using the resulting GP to model BNNs' residuals (Blight & Ott, 1975; O'Hagan, 1978; Qiu et al., 2020). Conceptually, we extend finite BNNs into infinite ones. The proposed kernel has two crucial properties: It has (i) negligible values around the origin and (ii) cubic variance growth w.r.t. α. Using (i), we can approximately decompose the resulting *a posteriori* function output simply as *a posteriori* BNNs' output plus *a priori* the GP's output—our extension can therefore

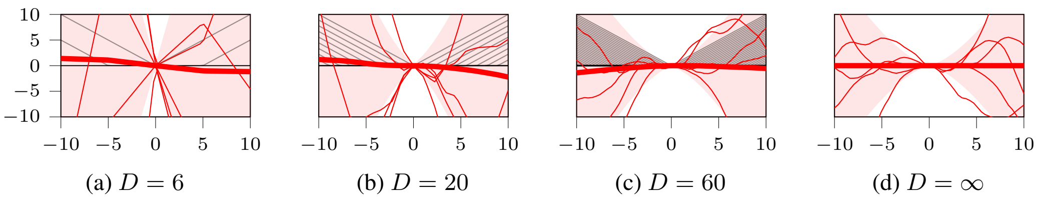


Figure 1: The construction of a GP prior with the proposed "ReLU kernel", as the limiting output covariance of a Bayesian linear model with D ReLU features, arranged at regular intervals, oriented away from the origin. With finite D, the variance grows quadratically, leading to the asymptotic overconfidence in ReLU BNNs. In contrast, with $D = \infty$, the variance grows *cubically*.

be applied to any pre-trained ReLU BNN in a *post-hoc* manner. And due to (ii), the extended BNNs exhibit super-quadratic output variance growth, and thus are guaranteed to predict with uniform confidence away from the data. This approach thus fixes ReLU BNNs' asymptotic overconfidence, without affecting the BNNs' predictive mean. The method can also be extended further while still preserving all these properties, by also modeling the representations of input points with the proposed GP such that it also improve the extended ReLU BNNs' non-asymptotic uncertainty.

2 Infinite-Feature Extension for ReLU BNNs

2.1 The Double-Sided Cubic Spline Kernel

The cubic spline kernel, obtained by placing infinitely many ReLU functions that points to the right on the (1D) input space and denoted by $\overrightarrow{k}^1(x, x'; \sigma^2)$ with a positive hyperparameter σ^2, is one-sided in the sense that it induces zero variance on $(-\infty, 0)$, and therefore is unsuitable for modeling over the entire domain (see Figure 2). We fix this issue by constructing another kernel with infinitely many ReLU functions pointing to the opposite direction $\overleftarrow{k}^1(x, x'; \sigma^2) := \overrightarrow{k}^1(-x, -x'; \sigma^2)$, which is non-zero only on $(-\infty, 0)$. Combining them together, we obtain the following kernel, which covers the whole real line $k^1(x, x'; \sigma^2) := \overleftarrow{k}^1(x, x'; \sigma^2) + \overrightarrow{k}^1(x, x'; \sigma^2)$, cf. Figure 1.

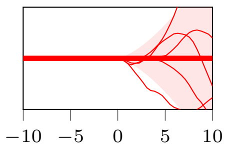


Figure 2: The GP prior induced by the cubic spline kernel.

For multivariate input domains, we define $k(\boldsymbol{x}, \boldsymbol{x}'; \sigma^2) := \frac{1}{N} \sum_{i=1}^{N} k^1(x_i, x_i'; \sigma^2)$ for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^N$ with $N > 1$. We call this kernel the ***double-sided cubic spline (DSCS) kernel***. Two crucial properties of this kernel are that (i) it has negligible variance around the origin of $\mathbb{R}^N$ and (ii) for any $\boldsymbol{x}_* \in \mathbb{R}^N$ and $\alpha \in \mathbb{R}$, the value $k(\alpha\boldsymbol{x}_*, \alpha\boldsymbol{x}_*)$ is *cubic* in α.

2.2 ReLU-GP Residual

Let $f : \mathbb{R}^N \times \mathbb{R}^D \to \mathbb{R}$ be an L-layer, real-valued ReLU BNN. Our goal is to extend f with the GP prior arising from the DSCS kernel to model its residual. We therefore place infinitely many ReLU features over its input space by following the construction in the previous section. Then, we arrive at a zero-mean GP prior $\mathcal{GP}(\widehat{f} \mid 0, k)$ over a real-valued random function $\widehat{f} : \mathbb{R}^N \to \mathbb{R}$ on the input space $\mathbb{R}^N$. Following previous works (Wahba, 1978; O'Hagan, 1978; Qiu et al., 2020), we use this GP prior to model the residual of f by

$$\widetilde{f} := f + \widehat{f}, \quad \text{where } \widehat{f} \sim \mathcal{GP}(0, k), \tag{1}$$

and call this method ***ReLU-GP residual (RGPR)***. Unlike these previous works, however, RGPR does not require expensive GP posterior inference since intuitively, the additional infinitely many ReLU features are never part of the training process because they are pointing away from the data. The following proposition, which uses property (i) of the DSCS kernel, formalizes this intuition under the linearization of f, assuming w.l.o.g. that the data are sufficiently close to the origin.

Proposition 1. *Let $f : \mathbb{R}^N \times \mathbb{R}^D \to \mathbb{R}$ be a ReLU regression BNN with prior $\mathcal{N}(\boldsymbol{\theta} \mid \mathbf{0}, \boldsymbol{B})$ satisfying $f_{\boldsymbol{\theta}=\mathbf{0}} \equiv 0$, $\mathcal{D} := (\boldsymbol{x}_m, y_m)_{m=1}^M$ a dataset, $\nu > 0$ an observation noise variance for $\widehat{f}$, and $\boldsymbol{x}_* \in \mathbb{R}^N$ an arbitrary input point. If $k(\boldsymbol{x}_m, \boldsymbol{x}) \approx 0$ for any $\boldsymbol{x} \in \mathbb{R}^N$ and for all $m = 1, \dots, M$, then under the linearization of f w.r.t. $\boldsymbol{\theta}$ around $\mathbf{0}$, the posterior over $\widetilde{f}_*$ is given by $p(\widetilde{f}_* \mid \boldsymbol{x}_*, \mathcal{D}) \approx \mathcal{N}(\boldsymbol{\mu}^\top \boldsymbol{g}_*, \boldsymbol{g}_*^\top \boldsymbol{\Sigma} \boldsymbol{g}_* + k_*)$, where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the parameters of the approximate posterior of f, and $\boldsymbol{g}_* := \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_*)|_{\mathbf{0}}$.*

In the result above, we notice that $\boldsymbol{\mu}^\top \boldsymbol{g}_*$ and $\boldsymbol{g}_*^\top \boldsymbol{\Sigma} \boldsymbol{g}_*$ are the mean and covariance of the posterior predictive distribution of the linearized network. Hence, we can think of the above posterior as the distribution of the sum of random variables $f_* \sim \mathcal{N}(\boldsymbol{\mu}^\top \boldsymbol{g}_*, \boldsymbol{g}_*^\top \boldsymbol{\Sigma} \boldsymbol{g}_*)$ and $\widehat{f}_* \sim \mathcal{N}(0, k_*)$, the latter being the marginal GP *prior* (1) at $\boldsymbol{x}_*$. That is, we can approximately write *a posteriori* $\widetilde{f}$ as *a posteriori* f plus *a priori* $\widehat{f}$. RGPR can thus be applied in a *post-hoc* manner, irrespective to the training process of the BNN. Furthermore, Proposition 1 also shows that RGPR models only the *uncertainty residual* of the BNN since it only affects the predictive variance. This kind of simple *post-hoc* form is not true in general, and arises specifically with ReLU features under DSCS kernel.

Generalization to BNNs with multiple outputs is straightforward. Let $f : \mathbb{R}^N \times \mathbb{R}^D \to \mathbb{R}^C$ be a vector-valued, pre-trained, L-layer ReLU BNN with posterior $\mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$. We assume that the following sequence of real-valued random functions $(\widehat{f}^{(c)} : \mathbb{R}^{\widehat{N}} \to \mathbb{R})_{c=1}^C$ is independent and identically distributed as the GP prior (1), i.e. for all $c = 1, \dots, C$, the function $\widehat{f}^{(c)}$ is distributed as $\mathcal{GP}(0, k)$. Thus, for any $\boldsymbol{x}_* \in \mathbb{R}^N$, defining $\widehat{f}_* := (\widehat{f}_*^{(1)}, \dots, \widehat{f}_*^{(C)})^\top$, we have $p(\widehat{f}_*) = \mathcal{N}(\mathbf{0}, k_* \boldsymbol{I})$. So, under the linearization of f, the result for real-valued networks above implies that the marginal GP posterior of RGPR is approximately given by a Gaussian with mean and covariance

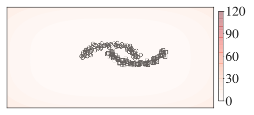
$$\mathbb{E}(\widetilde{f}_*) = f_{\boldsymbol{\mu}}(\boldsymbol{x}_*), \qquad \text{and} \qquad \mathrm{Cov}(\widetilde{f}_*) \; = \boldsymbol{J}_*^\top \boldsymbol{\Sigma} \boldsymbol{J}_* + k_* \boldsymbol{I}, \tag{2}$$

respectively. Note that in practice one is not required to use linearization—RGPR can be used with Monte Carlo integration, see Algorithm 1. Finally, we present our main result.

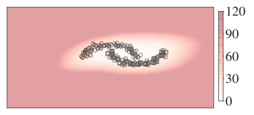
Theorem 2 (Uniform Asymptotic Confidence). *Let $f : \mathbb{R}^N \times \mathbb{R}^D \to \mathbb{R}^C$ be a C-class ReLU network equipped with the posterior $\mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ and let $\widetilde{f}$ be obtained from f via RGPR. Suppose that the linearization of f and the generalized probit approximation (Gibbs, 1997) is used for approximating the predictive distribution $p(y_* = c \mid \alpha \boldsymbol{x}_*, \widetilde{f}, \mathcal{D})$ under $\widetilde{f}$. For any input $\boldsymbol{x}_* \in \mathbb{R}^N$ with $\boldsymbol{x}_* \neq \mathbf{0}$ and for every class $c = 1, \dots, C$ we have $\lim_{\alpha \to \infty} p(y_* = c \mid \alpha \boldsymbol{x}_*, \widetilde{f}, \mathcal{D}) = 1/C$.*

2.3 Extending RGPR to Non-Asymptotic Regimes

While the previous construction is sufficient for modeling uncertainty far away from the data, it does not necessarily model the uncertainty *near* the data region well (Figure 3(a)). A way to address this is to adapt RGPR's notion of proximity between input points, by using the higher-level data representations already available from the pre-trained NN—a test point close to the data in the input space can be far from them in the representation space, thereby the DSCS kernel might assign a large variance. We therefore extend RGPR by additionally placing infinite ReLU features on the representation spaces of the point-estimated network $f_{\boldsymbol{\mu}}$ induced by the BNN f, where $\boldsymbol{\mu}$ is the mean of posterior of f.



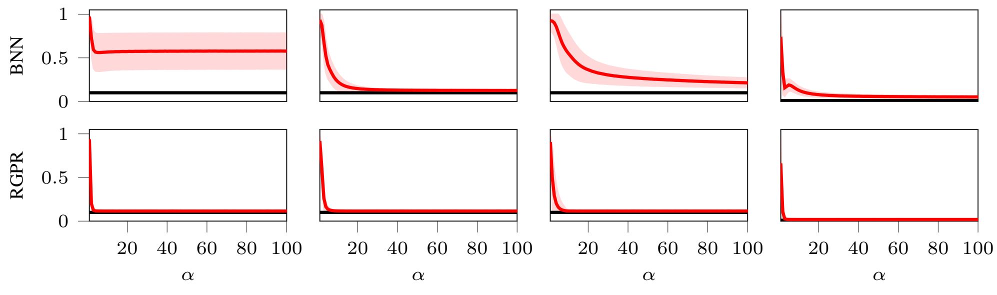
(a) Input Only



(b) Input & Hiddens

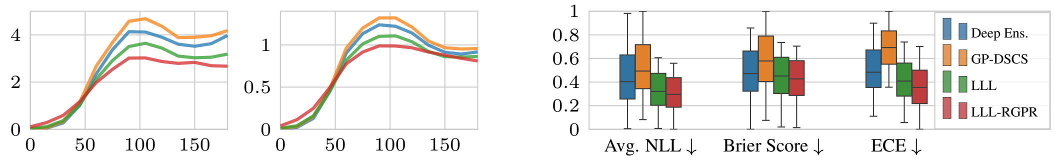
Figure 3: Variance of $\widehat{f}$.

For each $l = 1, \dots, L-1$ and any input $\boldsymbol{x}_*$, let N_l be the size of the l-th hidden layer of $f_{\boldsymbol{\mu}}$ and $\boldsymbol{h}_*^{(l)}$ be the l-th hidden representation of $\boldsymbol{x}_*$. By convention, we assume that $N_0 := N$ and $\boldsymbol{h}_*^{(0)} := \boldsymbol{x}_*$. We place an infinite number of ReLU features on the representation space $\mathbb{R}^{N_l}$, and thus we obtain a random function $\widehat{f}^{(l)} : \mathbb{R}^{N_l} \to \mathbb{R}$ distributed as the Gaussian process $\mathcal{GP}(0, k)$. Then, given that $\widehat{N} := \sum_{l=0}^{L-1} N_l$, we define the function $\widehat{f} : \mathbb{R}^{\widehat{N}} \to \mathbb{R}$ by $\widehat{f} := \sum_{l=0}^{L-1} \widehat{f}^{(l)}$. This function is therefore a function over *all* representation (including the input) spaces of $f_{\boldsymbol{\mu}}$, distributed as the additive Gaussian process $\mathcal{GP}(0, \sum_{l=0}^{L-1} k)$. In other words, given all representations $\boldsymbol{h}_* := (\boldsymbol{h}_*^{(l)})_{l=0}^{L-1}$ of $\boldsymbol{x}_*$ under $f_{\boldsymbol{\mu}}$, the marginal over the function output $\widehat{f}(\boldsymbol{h}_*)$ is given by



(a) MNIST (b) CIFAR10 (c) SVHN (d) CIFAR100

Figure 4: Confidence of a vanilla BNN (**top**) and the same BNN with RGPR (**bottom**), as a function of α. Test data are constructed by scaling the original test set with α. Red curves are means, shades are ± 1 standard deviations. Black lines are the ideal uniform confidences.



(a) Avg. NLL ↓ (b) Brier Score ↓ (c) CIFAR-10-C

Figure 5: **(a-b)** Rotated-MNIST results (averages of 10 prediction runs): x-axes are rotation angles. **(c)** Corrupted-CIFAR10 results: values are normalized to $[0, 1]$; box represents quartiles and error bar shows min. and max. values, over all types of corruption and all severity levels.

$p(\widehat{f}_*) = \mathcal{N}\left(0, \sum_{l=0}^{L-1} k\left(\boldsymbol{h}_*^{(l)}, \boldsymbol{h}_*^{(l)}; \sigma_l^2\right)\right)$. We can then use this definition of $\widehat{f}$ as a drop-in replacement in (1) to define RGPR. Figure 3(b) visualizes the effect: the low-variance region modeled by the random function $\widehat{f}$ becomes more compact around the data.

3 Empirical Evaluations

First, we validate Theorem 2. We use the last-layer Laplace (LLL) as the base BNN (Kristiadi et al., 2020). Results with other BNNs are in Appendix E. We show in Figure 4 the confidence estimates of both the BNN and the RGPR-imbued BNN over 1000 samples obtained from each of MNIST, CIFAR10, SVHN, and CIFAR100 test sets, as the scaling factor α increases. The vanilla BNN does not achieve the ideal uniform confidence prediction, even for large α. RGPR fixes this issue: the BNN's confidence estimates now converge to the uniform confidence. Moreover, these convergences happen at some finite, small α.

To validate RGPR on non-asymptotic regimes, we report results on standard dataset shift and out-of-distribution (OOD) detection tasks. We use LLL as the base BNN for RGPR and compare it against the MAP-trained network, the method of Qiu et al. (2020) with the DSCS kernel (GP-DSCS, see Appendix D), and Deep Ensemble (DE, Lakshminarayanan et al., 2017). First, for the dataset-shift task, we observe in Figure 5 that RGPR consistently improves the base LLL. A similar observation can also be seen in OOD detection (Table 1): RGPR can improve LLL further, making it better than DE.

Table 1: OOD data detection in terms of FPR@95 (%). Values are averages over five OOD test sets.

Methods	**MNIST**	**CIFAR10**	**SVHN**	**CIFAR100**
MAP	28.2	38.9	17.9	72.2
Deep Ens.	23.0	51.1	11.4	74.9
GP-DSCS	27.8	46.7	19.2	69.2
LLL	24.8	29.9	15.9	69.4
LLL-RGPR	**5.1**	**27.4**	**8.9**	**63.4**

4 Conclusion

We have shown that extending finite ReLU BNNs with an infinite set of additional, carefully placed ReLU features fixes their asymptotic overconfidence. The simplicity of our method is its main strength: RGPR causes no additional overhead during BNNs' training, but nevertheless meaningfully approximates a full GP posterior. The intuition behind RGPR is relatively simple, but it bridges the worlds of deep learning and non-parametric/kernel models: Correctly modeling uncertainty across the input domain requires a non-parametric model of infinitely many ReLU features, but only finitely many such features need to be trained to make good point predictions.

References

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding Deep Neural Networks with Rectified Linear Units. In *ICLR*, 2018.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

BJN Blight and L Ott. A Bayesian Approach to Model Inadequacy for Polynomial Regression. *Biometrika*, 62, 1975.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

Andrew YK Foong, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. 'in-between'uncertainty in bayesian neural networks. *arXiv*, 2019.

Mark N Gibbs. Bayesian Gaussian Processes for Regression and Classification. *Ph. D. Thesis, Department of Physics, University of Cambridge*, 1997.

Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why ReLU Networks Yield High-confidence Predictions Far Away from the Training Data and How to Mitigate the Problem. In *CVPR*, 2019.

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep Anomaly Detection with Outlier Exposure. In *ICLR*, 2019.

Geoffrey E Hinton and Drew Van Camp. Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights. In *COLT*, 1993.

Alexander Immer, Maciej Korzepa, and Matthias Bauer. Improving predictions of bayesian neural networks via local linearization. *arXiv preprint arXiv:2008.08400*, 2020.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks. In *ICML*, 2020.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *NIPS*, 2017.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples. In *ICLR*, 2018.

Zhiyun Lu, Eugene Ie, and Fei Sha. Uncertainty estimation with infinitesimal jackknife, its distribution and mean-field approximation. *arXiv preprint arXiv:2006.07584*, 2020.

David JC MacKay. The Evidence Framework Applied to Classification Networks. *Neural computation*, 1992a.

David JC MacKay. A Practical Bayesian Framework For Backpropagation Networks. *Neural computation*, 4(3), 1992b.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A Simple Baseline for Bayesian Uncertainty in Deep Learning. In *NeurIPS*, 2019.

Vinod Nair and Geoffrey E Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML*, 2010.

Anthony O'Hagan. Curve Fitting and Optimal Design for Prediction. *Journal of the Royal Statistical Society: Series B (Methodological)*, 40, 1978.

Xin Qiu, Elliot Meyerson, and Risto Miikkulainen. Quantifying Point-Prediction Uncertainty in Neural Networks via Residual Estimation with an I/O Kernel. In *ICLR*, 2020.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes in Machine Learning*. 2005.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A Scalable Laplace Approximation for Neural Networks. In *ICLR*, 2018.

Grace Wahba. Improper Priors, Spline Smoothing and the Problem of Guarding Against Model Errors in Regression. *Journal of the Royal Statistical Society: Series B (Methodological)*, 40, 1978.

Grace Wahba. *Spline Models for Observational Data*. SIAM, 1990.

Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic Variational Deep Kernel Learning. In *NIPS*, 2016.

APPENDIX A ADDITIONAL BACKGROUND

We denote a test point and its unknown label as $\boldsymbol{x}_*$ and y_*, respectively. Furthermore, we denote any quantity that depends on $\boldsymbol{x}_*$ with the same subscript. E.g., we write $f_* := f(\boldsymbol{x}_*)$, $k_* := k(\boldsymbol{x}_*, \boldsymbol{x}_*)$, and $\boldsymbol{m}_* := \boldsymbol{m}(\boldsymbol{x}_*)$.

A.1 BAYESIAN NEURAL NETWORKS

We focus on multi-class classification problems. Let $f : \mathbb{R}^N \times \mathbb{R}^D \to \mathbb{R}^C$ defined by $(\boldsymbol{x}, \boldsymbol{\theta}) \mapsto f(\boldsymbol{x}; \boldsymbol{\theta}) =: f_{\boldsymbol{\theta}}(\boldsymbol{x})$ be a C-class ReLU neural network—a fully-connected or convolutional feed-forward network equipped with the ReLU nonlinearity. Here, $\boldsymbol{\theta}$ is the collection of all parameters of f. Given an i.i.d. dataset $\mathcal{D} := (\boldsymbol{x}_m, y_m)_{m=1}^M$, the standard training procedure amounts to finding a ***maximum a posteriori (MAP) estimate*** $\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} \mid \mathcal{D})$.

One can also treat $\boldsymbol{\theta}$ as a random variable and apply Bayes' theorem—the resulting network is called a ***Bayesian neural network (BNN)***. A common way to approximate the posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$ of a BNN is by a Gaussian $q(\boldsymbol{\theta} \mid \mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, which can be constructed for example via a Laplace or variational approximation (MacKay, 1992b; Hinton & Van Camp, 1993). Given an approximate posterior and a test point $\boldsymbol{x}_* \in \mathbb{R}^N$, one then needs to marginalize the parameters to make predictions, i.e. we compute

$$p(y_* \mid \boldsymbol{x}_*, \mathcal{D}) = \int p(y_* \mid f_{\boldsymbol{\theta}}(\boldsymbol{x}_*))\, q(\boldsymbol{\theta} \mid \mathcal{D})\, d\boldsymbol{\theta},$$

where $p(y_* \mid f_{\boldsymbol{\theta}}(\boldsymbol{x}_*))$ is the likelihood, often chosen as the softmax function. However, since the network f is a non-linear function of $\boldsymbol{\theta}$, this integral does not have an analytic solution. But one can obtain a useful closed-form approximation as follows. First, we perform a ***network linearization*** on f around $\boldsymbol{\mu}$ and obtain the following distribution over the function output $f(\boldsymbol{x}_*)$, where $\boldsymbol{\theta}$ has been marginalized:[1]

$$p(f_* \mid \boldsymbol{x}_*, \mathcal{D}) \approx \mathcal{N}(\underbrace{f_{\boldsymbol{\mu}}(\boldsymbol{x}_*)}_{=:\boldsymbol{m}_*}, \underbrace{\boldsymbol{J}_*^\top \boldsymbol{\Sigma} \boldsymbol{J}_*}_{=:\boldsymbol{V}_*}). \tag{3}$$

Here, $\boldsymbol{J}_*$ is the $D \times C$ Jacobian matrix of $f_{\boldsymbol{\theta}}(\boldsymbol{x}_*)$ w.r.t. $\boldsymbol{\theta}$ at $\boldsymbol{\mu}$. On top of that, we apply the ***generalized probit approximation***, due to Gibbs (1997); MacKay (1992a):

$$p(y_* = c \mid \boldsymbol{x}_*, \mathcal{D}) \approx \frac{\exp(m_{*c}\, \kappa_{*c})}{\sum_{i=1}^C \exp(m_{*i}\, \kappa_{*i})}, \tag{4}$$

where for each $i = 1, \dots, C$, the real number m_{*i} is the i-th component of the vector $\boldsymbol{m}_*$, and $\kappa_{*i} := (1 + \pi/8\, v_{*ii})^{-1/2}$ where v_{*ii} is the i-th diagonal term of the matrix $\boldsymbol{V}_*$. Both approximations have been shown to be particularly good both in terms of their approximation errors and predictive performance (MacKay, 1992a; Foong et al., 2019; Immer et al., 2020; Lu et al., 2020).

While analytically useful, these approximations can be expensive due to the computation of the Jacobian matrix $\boldsymbol{J}_*$. Thus, ***Monte Carlo (MC) integration*** is commonly used as an alternative, i.e. we approximate

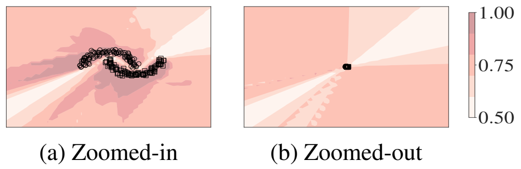
$$p(y_* \mid \boldsymbol{x}_*, \mathcal{D}) \approx \frac{1}{S} \sum_{s=1}^S p(y_* \mid f_{\boldsymbol{\theta}_s}(\boldsymbol{x}_*)); \;\; \boldsymbol{\theta}_s \sim q(\boldsymbol{\theta} \mid \mathcal{D}).$$

Given a classification predictive distribution $p(y_* \mid \boldsymbol{x}_*, \mathcal{D})$, we define the predictive ***confidence*** of $\boldsymbol{x}_*$ as the maximum probability $\text{conf}(\boldsymbol{x}_*) := \max_{c \in \{1,\dots,C\}} p(y_* = c \mid \boldsymbol{x}_*, \mathcal{D})$ over class labels. Far from the data, a well-calibrated model should produce the ***uniform confidence*** of $1/C$ or equivalently $p(y_* = c \mid \boldsymbol{x}_*, \mathcal{D}) = 1/C$ for all $c = 1, \dots, C$.

A.2 ASYMPTOTIC OVERCONFIDENCE IN BNNS

Given a fixed point estimate $\boldsymbol{\theta}_{\text{MAP}}$, the ReLU network $f_{\boldsymbol{\theta}_{\text{MAP}}}$ yields overconfident predictions, even for points far away from the training data (Hein et al., 2019). That is, for almost any input $\boldsymbol{x}_* \in \mathbb{R}^N$,

[1]See Bishop (2006, Sec. 5.7.1) for more details.



(a) Zoomed-in (b) Zoomed-out

Figure 6: Toy classification with a BNN. Shades represent confidence. Far from data, while its confidence estimates are bounded away from 1, BNN can still produce overconfident predictions.

one can show that there exists a class $c \in \{1, \dots, C\}$ such that $\lim_{\alpha\to\infty} \operatorname{softmax}(f_{\boldsymbol{\theta}_{\text{MAP}}}(\alpha \boldsymbol{x}_*))_c = 1$. Intuitively, this issue arises because the ReLU network yields a piecewise-affine function with finitely many linear regions (the domain of each affine function). Under this setup, by scaling $\boldsymbol{x}_*$ with α, at some point one arrives at an "outer linear region" and in this region, the network is always affine—either increasing or decreasing—and thus its softmax output converges to a "one-hot vector" as α tends to infinity.

BNNs, even with a simple Gaussian approximate posterior $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, can help to mitigate this problem in binary classifications (Kristiadi et al., 2020). The crux of their proof is the observation that in an outer linear region, the predictive distribution (via the probit approximation) is given by[2]

$$p(y_* = 1 \mid \alpha \boldsymbol{x}_*, \mathcal{D}) = \sigma\left(\frac{\alpha \boldsymbol{u}^\top \boldsymbol{x}_*}{\sqrt{1 + \pi/8\, v(\alpha \boldsymbol{x}_*)}}\right), \tag{5}$$

where σ is the logistic-sigmoid function, $\boldsymbol{u}$ is the parameter vector corresponding to the linear region, and v maps $\alpha \boldsymbol{x}_*$ to the variance of the network output and is a quadratic function of α. Unfortunately, both the numerator and denominator above are linear in α and thus altogether (5) only converges to a constant as $\alpha \to \infty$, not necessarily the ideal uniform confidence prediction. BNNs can therefore still be overconfident, albeit less so than the point-estimated counterpart. We refer the reader to Figure 6 for an intuition.

A.3 RELU AND GAUSSIAN PROCESSES

The ReLU activation function $\operatorname{ReLU}(z) := \max(0, z)$ (Nair & Hinton, 2010) has become the *de-facto* choice of non-linearity in deep learning. Given an arbitrary real number c, it can be generalized as $\operatorname{ReLU}(z; c) := \max(0, z - c)$, with the "kink" at location c. An alternative formulation, useful below, is in terms of the Heaviside function H as $\operatorname{ReLU}(z; c) = H(z - c) \cdot (z - c)$. We may define a collection of K such ReLU functions evaluated at some point in $\mathbb{R}$ as the function $\boldsymbol{\phi} : \mathbb{R} \to \mathbb{R}^K$ with $z \mapsto (\operatorname{ReLU}(z; c_1), \dots, \operatorname{ReLU}(z; c_K))^\top$. We call this function the ***ReLU feature map***, which can be interpreted as "placing" ReLU functions at different locations in $\mathbb{R}$.

Consider a linear model $g : \mathbb{R} \times \mathbb{R}^K \to \mathbb{R}$ defined by $g(x; \boldsymbol{w}) := \boldsymbol{w}^\top \boldsymbol{\phi}(x)$. Suppose $\boldsymbol{\phi}$ regularly places the K generalized ReLU functions centered at $(c_i)_{i=1}^K$ over $[c_{\text{min}}, c_{\text{max}}] \subset \mathbb{R}$, where $c_{\text{min}} < c_{\text{max}}$. If we consider a Gaussian prior $p(\boldsymbol{w}) := \mathcal{N}\left(\boldsymbol{w} \,\middle|\, \boldsymbol{0}, \sigma^2 K^{-1}(c_{\text{max}} - c_{\text{min}})\boldsymbol{I}\right)$ over the weights $\boldsymbol{w}$ then, as K goes to infinity, the distribution over g is a Gaussian process with mean 0 and covariance[3]

$$\begin{aligned} \widehat{k}^1(x, x'; c_{\text{min}}, \sigma^2) &:= \lim_{K\to\infty} \operatorname{cov}(g(x), g(x')) \\ &= \sigma^2 H(\bar{x} - c_{\text{min}}) \left(\frac{1}{3}(\bar{x}^3 - c_{\text{min}}^3) - \frac{1}{2}(\bar{x}^2 - c_{\text{min}}^2)(x + x') + (\bar{x} - c_{\text{min}})xx'\right). \end{aligned} \tag{6}$$

Here, the superscript 1 denotes the fact that this function is over a 1-dimensional input space and $\bar{x} := \min(x, x')$. Since the expression above does not depend on c_{max}, we can consider the limit $c_{\text{max}} \to \infty$, and thus this kernel is non-zero on (c_{min}, ∞). This covariance function is the ***cubic spline kernel*** (Wahba, 1990). The name indicates that posterior mean function of the associated GP is piecewise-cubic. But it also has the property that the variance $\widehat{k}^1(x, x; c_{\text{min}}, \sigma^2)$ grows cubically in x and this variance is negligible for x close to c_{min}. When used as a GP prior with $c_{\text{min}} = 0$ and $\sigma^2 = 1$, the resulting GP is the one in Figure 1(d), but with zero variance over $(-\infty, 0]$.

[2]We omit the bias parameter for simplicity.

[3]Full derivation in Appendix B.

APPENDIX B DERIVATIONS

B.1 THE CUBIC SPLINE KERNEL

Recall that we have a linear model $f : [c_{\text{min}}, c_{\text{max}}] \times \mathbb{R}^K \to \mathbb{R}$ with the ReLU feature map $\boldsymbol{\phi}$ defined by $f(x; \boldsymbol{w}) := \boldsymbol{w}^\top \boldsymbol{\phi}(x)$ over the input space $[c_{\text{min}}, c_{\text{max}}] \subset \mathbb{R}$, where $c_{\text{min}} < c_{\text{max}}$. Furthermore, $\boldsymbol{\phi}$ regularly places the K generalized ReLU functions centered at $(c_i)_{i=1}^K$ where $c_i = c_{\min} + \frac{i-1}{K-1}(c_{\max} - c_{\min})$ in the input space, and we consider a Gaussian prior $p(\boldsymbol{w}) := \mathcal{N}\left(\boldsymbol{w} \,\middle|\, \boldsymbol{0}, \sigma^2 K^{-1}(c_{\text{max}} - c_{\text{min}})\boldsymbol{I}\right)$ over the weight $\boldsymbol{w}$. Then, as K goes to infinity, the distribution over the function output $f(x)$ is a Gaussian process with mean 0 and covariance

$$\begin{aligned}\operatorname{cov}(f(x), f(x')) &= \sigma^2 \frac{c_{\text{max}} - c_{\text{min}}}{K} \boldsymbol{\phi}(x)^\top \boldsymbol{\phi}(x') = \sigma^2 \frac{c_{\text{max}} - c_{\text{min}}}{K} \sum_{i=1}^K \operatorname{ReLU}(x; c_i) \operatorname{ReLU}(x'; c_i) \\ &= \sigma^2 \frac{c_{\text{max}} - c_{\text{min}}}{K} \sum_{i=1}^K H(x - c_i) H(x' - c_i)(x - c_i)(x' - c_i) \\ &= \sigma^2 \frac{c_{\text{max}} - c_{\text{min}}}{K} \sum_{i=1}^K H(\min(x, x') - c_i) \left(c_i^2 - c_i(x + x') + xx'\right), \qquad (7)\end{aligned}$$

where the last equality follows from (i) the fact that both x and x' must be greater than or equal to c_i, and (ii) by expanding the quadratic form in the second line.

Let $\bar{x} := \min(x, x')$. Since (7) is a Riemann sum, in the limit of $K \to \infty$, it is expressed by the following integral

$$\begin{aligned}\lim_{K\to\infty} \operatorname{cov}(f(x), f(x')) &= \sigma^2 \int_{c_{\text{min}}}^{c_{\text{max}}} H(\bar{x} - c) \left(c^2 - c(x + x') + xx'\right) \, dc \\ &= \sigma^2 H(\bar{x} - c_{\text{min}}) \int_{c_{\text{min}}}^{\min\{\bar{x}, c_{\max}\}} c^2 - c(x + x') + xx' \, dc \\ &= \sigma^2 H(\bar{x} - c_{\text{min}}) \left[\frac{1}{3}(z^3 - c_{\text{min}}^3) - \frac{1}{2}(z^2 - c_{\text{min}}^2)(x + x') + (z - c_{\text{min}})xx'\right]\end{aligned}$$

where we have defined $z := \min\{\bar{x}, c_{\max}\}$. The term $H(\bar{x} - c_{\text{min}})$ has been added in the second equality as the previous expression is zero if $\bar{x} \leq c_{\text{min}}$ (since in this region, all the ReLU functions evaluate to zero). Note that

$$H(\bar{x} - c_{\text{min}}) = H(x - c_{\text{min}}) H(x' - c_{\text{min}})$$

is itself a positive definite kernel. We also note that c_{max} can be chosen sufficiently large so that $[-c_{\text{max}}, c_{\text{max}}]^d$ contains for sure the data, e.g. this is anyway true for data from bounded domains like images in $[0, 1]^d$, and thus we can set $z = \bar{x} = \min(x, x')$.

APPENDIX C PROOFS

Proposition 1. *Let $f : \mathbb{R}^N \times \mathbb{R}^D \to \mathbb{R}$ be a ReLU regression BNN with prior $\mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{0}, \boldsymbol{B})$ satisfying $f_{\boldsymbol{\theta}=\boldsymbol{0}} \equiv 0$, $\mathcal{D} := (\boldsymbol{x}_m, y_m)_{m=1}^M$ a dataset, $\nu > 0$ an observation noise variance for $\widehat{f}$, and $\boldsymbol{x}_* \in \mathbb{R}^N$ an arbitrary input point. If $k(\boldsymbol{x}_m, \boldsymbol{x}) \approx 0$ for any $\boldsymbol{x} \in \mathbb{R}^N$ and for all $m = 1, \ldots, M$, then under the linearization of f w.r.t. $\boldsymbol{\theta}$ around $\boldsymbol{0}$, the posterior over $\widetilde{f}_*$ is given by $p(\widetilde{f}_* \mid \boldsymbol{x}_*, \mathcal{D}) \approx \mathcal{N}(\boldsymbol{\mu}^\top \boldsymbol{g}_*, \boldsymbol{g}_*^\top \boldsymbol{\Sigma} \boldsymbol{g}_* + k_*)$, where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the parameters of the approximate posterior of f, and $\boldsymbol{g}_* := \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_*)|_{\boldsymbol{0}}$.*

Proof. Under the linearization of f w.r.t. $\boldsymbol{\theta}$ around $\boldsymbol{0}$, we have

$$f(\boldsymbol{x}; \boldsymbol{\theta}) \approx \underbrace{f(\boldsymbol{x}; \boldsymbol{0})}_{=0} + \underbrace{\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}; \boldsymbol{\theta})|_{\boldsymbol{0}}}_{=: \boldsymbol{g}(\boldsymbol{x})}{}^\top \boldsymbol{\theta} = \boldsymbol{g}(\boldsymbol{x})^\top \boldsymbol{\theta}.$$

Along with the definition of RGPR, this implies that we have

$$\widetilde{f}(\boldsymbol{x}) \approx \boldsymbol{g}(\boldsymbol{x})^\top \boldsymbol{\theta} + \widehat{f}^{(0)}(\boldsymbol{x}); \qquad \widehat{f}^{(0)}(\boldsymbol{x}) \sim \mathcal{N}(0, k(\boldsymbol{x}, \boldsymbol{x})).$$

Following O'Hagan (1978), we thus obtain the following marginal GP prior over $\widetilde{f}$.

$$\widetilde{f}(\boldsymbol{x}) \sim \mathcal{N}(\widetilde{f}(\boldsymbol{x}) \mid 0, \boldsymbol{g}(\boldsymbol{x})^\top \boldsymbol{B} \boldsymbol{g}(\boldsymbol{x}) + k(\boldsymbol{x}, \boldsymbol{x})).$$

Suppose we write the dataset as $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$ where $\boldsymbol{X}$ is the data matrix and $\boldsymbol{y}$ is the target vectors, and $\boldsymbol{x}_* \in \mathbb{R}^N$ is an arbitrary test point. Let $\boldsymbol{k} := (k(\boldsymbol{x}_*, \boldsymbol{x}_1), \dots k(\boldsymbol{x}_*, \boldsymbol{x}_M))^\top$, let $\widehat{\boldsymbol{K}} := (\boldsymbol{K} + \nu \boldsymbol{I})$ be the kernel matrix under the observation noise ν with $K_{ij} := k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and let $\boldsymbol{G} := (\boldsymbol{g}(\boldsymbol{x}_1), \dots, \boldsymbol{g}(\boldsymbol{x}_M))$ be the matrix of training "features". Following notation and derivation in Rasmussen & Williams (2005, Sec. 2.7), we get the following GP posterior mean and variance

$$\mathbb{E}(\widetilde{f}_* \mid \mathcal{D}) = \boldsymbol{g}_*^\top \boldsymbol{\mu} + \boldsymbol{k} \widehat{\boldsymbol{K}}^{-1} (\boldsymbol{y} - \boldsymbol{g}_*^\top \boldsymbol{\mu}) \tag{8}$$

$$\operatorname{var}(\widetilde{f}_* \mid \mathcal{D}) = k_* + \boldsymbol{k}^\top \widehat{\boldsymbol{K}}^{-1} \boldsymbol{k} + \boldsymbol{r}^\top (\boldsymbol{B}^{-1} + \boldsymbol{G} \widehat{\boldsymbol{K}}^{-1} \boldsymbol{G}^\top)^{-1} \boldsymbol{r}, \tag{9}$$

where $\boldsymbol{\mu} := (\boldsymbol{B}^{-1} + \boldsymbol{G} \widehat{\boldsymbol{K}}^{-1} \boldsymbol{G}^\top)^{-1} \boldsymbol{G} \widehat{\boldsymbol{K}}^{-1} \boldsymbol{y}$ and $\boldsymbol{r} := \boldsymbol{g}_* - \boldsymbol{G} \widehat{\boldsymbol{K}}^{-1} \boldsymbol{k}$. By hypothesis, we have that $\boldsymbol{k} \approx \boldsymbol{0}$ and thus we also have $\widehat{\boldsymbol{K}}^{-1} \approx 1/\nu \boldsymbol{I}$. These imply that

$$\boldsymbol{\mu} \approx (\boldsymbol{B}^{-1} + 1/\nu \boldsymbol{G}\boldsymbol{G}^\top)^{-1} (1/\nu \boldsymbol{G} \boldsymbol{y}) \qquad \text{and} \qquad \boldsymbol{r} \approx \boldsymbol{g}_*.$$

In particular, notice that $\boldsymbol{\mu}$ is approximately the posterior mean of the Bayesian linear regression on f (Bishop, 2006, Sec. 3.3). Furthermore, (8) and (9) become

$$\begin{aligned} \mathbb{E}(\widetilde{f}_* \mid \mathcal{D}) &\approx \boldsymbol{\mu}^\top \boldsymbol{g}_* \\ \operatorname{var}(\widetilde{f}_* \mid \mathcal{D}) &\approx k_* + \boldsymbol{g}(\boldsymbol{x}_*)^\top \underbrace{(\boldsymbol{B}^{-1} + 1/\nu \boldsymbol{G}\boldsymbol{G}^\top)^{-1}}_{=:\boldsymbol{\Sigma}} \boldsymbol{g}_*, \end{aligned}$$

respectively. Notice in particular that $\boldsymbol{\Sigma}$ is the corresponding posterior covariance of the Bayesian linear regression on f. Thus, the claim follows. $\square$

To prove Lemma 4 and Theorem 2, we need the following definition. Let $f : \mathbb{R}^N \times \mathbb{R}^D \to \mathbb{R}^C$ defined by $(\boldsymbol{x}, \boldsymbol{\theta}) \mapsto f(\boldsymbol{x}; \boldsymbol{\theta})$ be a feed-forward neural network which uses piecewise affine activation functions (such as ReLU and leaky-ReLU) and are linear in the output layer. Such a network is called a ***ReLU network*** and can be written as a continuous piecewise-affine function (Arora et al., 2018). That is, there exists a finite set of polytopes $\{Q_i\}_{i=1}^P$—referred to as ***linear regions*** f—such that $\cup_{i=1}^P Q_i = \mathbb{R}^N$ and $f|_{Q_i}$ is an affine function for each $i = 1, \dots, P$ (Hein et al., 2019). The following lemma is central in our proofs below (the proof is in Lemma 3.1 of Hein et al. (2019)).

Lemma 3 (Hein et al., 2019). *Let $\{Q_i\}_{i=1}^P$ be the set of linear regions associated to the ReLU network $f : \mathbb{R}^N \times \mathbb{R}^D \to \mathbb{R}^C$, For any $\boldsymbol{x} \in \mathbb{R}^N$ with $\boldsymbol{x} \neq 0$ there exists a positive real number β and $j \in \{1, \dots, P\}$ such that $\alpha \boldsymbol{x} \in Q_j$ for all $\alpha \geq \beta$.* $\square$

Lemma 4. *Let $f : \mathbb{R}^N \times \mathbb{R}^D \to \mathbb{R}^C$ be a C-class ReLU network with posterior $\mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\widetilde{f}$ be obtained from f via RGPR. Suppose that the linearization of f w.r.t. $\boldsymbol{\theta}$ around $\boldsymbol{\mu}$ is employed. For any $\boldsymbol{x}_* \in \mathbb{R}^N$ with $\boldsymbol{x}_* \neq \boldsymbol{0}$ there exists $\beta > 0$ such that for any $\alpha \geq \beta$, the variance of each output component $\widetilde{f}^{(1)}(\alpha \boldsymbol{x}_*), \dots, \widetilde{f}^{(C)}(\alpha \boldsymbol{x}_*)$ under $p(\widetilde{f}_* \mid \boldsymbol{x}_*, \mathcal{D})$ (2) is in $\Theta(\alpha^3)$.*

Proof. Let $\boldsymbol{x}_* \in \mathbb{R}^N$ with $\boldsymbol{x}_* \neq \boldsymbol{0}$ be arbitrary. By Lemma 3 and definition of ReLU network, there exists a linear region R and real number $\beta > 0$ such that for any $\alpha \geq \beta$, the restriction of f to R can be written as

$$f|_R(\alpha \boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{W}(\alpha \boldsymbol{x}) + \boldsymbol{b},$$

for some matrix $\boldsymbol{W} \in \mathbb{R}^{C \times N}$ and vector $\boldsymbol{b} \in \mathbb{R}^C$, which are functions of the parameter $\boldsymbol{\theta}$, evaluated at $\boldsymbol{\mu}$. In particular, for each $c = 1, \dots, C$, the c-th output component of $f|_R$ can be written by

$$f_c|_R = \boldsymbol{w}_c^\top (\alpha \boldsymbol{x}) + b_c,$$

where $\boldsymbol{w}_c$ and b_c are the c-th row of $\boldsymbol{W}$ and $\boldsymbol{b}$, respectively.

Let $c \in \{1, \dots, C\}$ and let $\boldsymbol{j}_c(\alpha \boldsymbol{x}_*)$ be the c-th column of the Jacobian $\boldsymbol{J}(\alpha \boldsymbol{x}_*)$ as defined in (3). Then by definition of $p(\widetilde{f}_* \mid \boldsymbol{x}_*, \mathcal{D})$, the variance of $\widetilde{f}_c|_R(\alpha \boldsymbol{x}_*)$—the c-th diagonal entry of the covariance of $p(\widetilde{f}_* \mid \boldsymbol{x}_*, \mathcal{D})$—is given by

$$\operatorname{var}(\widetilde{f}_c|_R(\alpha \boldsymbol{x}_*)) = \boldsymbol{j}_c(\alpha \boldsymbol{x}_*)^\top \boldsymbol{\Sigma} \boldsymbol{j}_c(\alpha \boldsymbol{x}_*) + k(\alpha \boldsymbol{x}_*, \alpha \boldsymbol{x}_*).$$

Now, from the definition of the DSCS kernel, we have

$$\begin{aligned} k(\alpha \boldsymbol{x}_*, \alpha \boldsymbol{x}_*) &= \frac{1}{N} \sum_{i=1}^N k^1(\alpha x_{*i}, \alpha x_{*i}) \\ &= \frac{1}{N} \sum_{i=1}^N \alpha^3 \frac{\sigma^2}{3} x_{*i}^3 \\ &= \frac{\alpha^3}{N} \sum_{i=1}^N k^1(x_{*i}, x_{*i}) \\ &\in \Theta(\alpha^3). \end{aligned}$$

Furthermore, we have

$$\boldsymbol{j}_c(\alpha \boldsymbol{x}_*)^\top \boldsymbol{\Sigma} \boldsymbol{j}_c(\alpha \boldsymbol{x}_*) = \left(\alpha (\nabla_{\boldsymbol{\theta}} \boldsymbol{w}_c|_{\boldsymbol{\mu}})^\top \boldsymbol{x} + \nabla_{\boldsymbol{\theta}} b_c|_{\boldsymbol{\mu}}\right)^\top \boldsymbol{\Sigma} \left(\alpha (\nabla_{\boldsymbol{\theta}} \boldsymbol{w}_c|_{\boldsymbol{\mu}})^\top \boldsymbol{x} + \nabla_{\boldsymbol{\theta}} b_c|_{\boldsymbol{\mu}}\right).$$

Thus, $\boldsymbol{j}_c(\alpha \boldsymbol{x}_*)^\top \boldsymbol{\Sigma} \boldsymbol{j}_c(\alpha \boldsymbol{x}_*)$ is a quadratic function of α. Therefore, $\operatorname{var}(\widetilde{f}_c|_R(\alpha \boldsymbol{x}_*))$ is in $\Theta(\alpha^3)$. □

Theorem 2 (Uniform Asymptotic Confidence). *Let $f : \mathbb{R}^N \times \mathbb{R}^D \to \mathbb{R}^C$ be a C-class ReLU network equipped with the posterior $\mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ and let $\widetilde{f}$ be obtained from f via RGPR. Suppose that the linearization of f and the generalized probit approximation (Gibbs, 1997) is used for approximating the predictive distribution $p(y_* = c \mid \alpha \boldsymbol{x}_*, \widetilde{f}, \mathcal{D})$ under $\widetilde{f}$. For any input $\boldsymbol{x}_* \in \mathbb{R}^N$ with $\boldsymbol{x}_* \neq \boldsymbol{0}$ and for every class $c = 1, \dots, C$ we have $\lim_{\alpha \to \infty} p(y_* = c \mid \alpha \boldsymbol{x}_*, \widetilde{f}, \mathcal{D}) = 1/C$.*

Proof. Let $\boldsymbol{x}_* \neq \boldsymbol{0} \in \mathbb{R}^N$ be arbitrary. By Lemma 3 and definition of ReLU network, there exists a linear region R and real number $\beta > 0$ such that for any $\alpha \geq \beta$, the restriction of f to R can be written as

$$f|_R(\alpha \boldsymbol{x}) = \boldsymbol{W}(\alpha \boldsymbol{x}) + \boldsymbol{b},$$

where the matrix $\boldsymbol{W} \in \mathbb{R}^{C \times N}$ and vector $\boldsymbol{b} \in \mathbb{R}^C$ are functions of the parameter $\boldsymbol{\theta}$, evaluated at $\boldsymbol{\mu}$. Furthermore, for $i = 1, \dots, C$ we denote the i-th row and the i-th component of $\boldsymbol{W}$ and $\boldsymbol{b}$ as $\boldsymbol{w}_i$ and b_i, respectively. Under the linearization of f, the marginal distribution (2) over the output $\widetilde{f}(\alpha \boldsymbol{x})$ holds. Hence, under the generalized probit approximation, the predictive distribution restricted to R is given by

$$\begin{aligned} \widetilde{p}(y_* = c \mid \alpha \boldsymbol{x}_*, \mathcal{D}) &\approx \frac{\exp(m_c(\alpha \boldsymbol{x}_*)\, \kappa_c(\alpha \boldsymbol{x}_*))}{\sum_{i=1}^C \exp(m_i(\alpha \boldsymbol{x}_*)\, \kappa_i(\alpha \boldsymbol{x}_*))} \\ &= \frac{1}{1 + \sum_{i \neq c}^C \exp(\underbrace{m_i(\alpha \boldsymbol{x}_*)\, \kappa_i(\alpha \boldsymbol{x}_*) - m_c(\alpha \boldsymbol{x}_*)\, \kappa_c(\alpha \boldsymbol{x}_*)}_{=: z_{ic}(\alpha \boldsymbol{x}_*)})}, \end{aligned}$$

where for all $i = 1, \dots, C$,

$$m_i(\alpha \boldsymbol{x}_*) = f_i|_R(\alpha \boldsymbol{x}; \boldsymbol{\mu}) = \boldsymbol{w}_i^\top (\alpha \boldsymbol{x}) + b_i \in \mathbb{R},$$

and

$$\kappa_i(\alpha \boldsymbol{x}) = (1 + \pi/8\, (v_{ii}(\alpha \boldsymbol{x}_*) + k(\alpha \boldsymbol{x}_*, \alpha \boldsymbol{x}_*)))^{-\frac{1}{2}} \in \mathbb{R}_{>0}.$$

In particular, for all $i = 1, \dots, C$, note that $m(\alpha \boldsymbol{x}_*)_i \in \Theta(\alpha)$ and $\kappa(\alpha \boldsymbol{x})_i \in \Theta(1/\alpha^{\frac{3}{2}})$ since $v_{ii}(\alpha \boldsymbol{x}_*) + k(\alpha \boldsymbol{x}_*, \alpha \boldsymbol{x}_*)$ is in $\Theta(\alpha^3)$ by Lemma 4. Now, notice that for any $c = 1, \dots, C$ and any $i \in \{1, \dots, C\} \setminus \{c\}$, we have

$$\begin{aligned} z_{ic}(\alpha \boldsymbol{x}_*) &= (m_i(\alpha \boldsymbol{x}_*)\, \kappa_i(\alpha \boldsymbol{x}_*)) - (m_c(\alpha \boldsymbol{x}_*)\, \kappa_c(\alpha \boldsymbol{x}_*)) \\ &= (\underbrace{\kappa_i(\alpha \boldsymbol{x}_*)\, \boldsymbol{w}_i}_{\Theta\left(1/\alpha^{\frac{3}{2}}\right)} - \underbrace{\kappa_c(\alpha \boldsymbol{x}_*)\, \boldsymbol{w}_c}_{\Theta\left(1/\alpha^{\frac{3}{2}}\right)})^\top (\alpha \boldsymbol{x}_*) + \underbrace{\kappa_i(\alpha \boldsymbol{x}_*)\, b_i}_{\Theta\left(1/\alpha^{\frac{3}{2}}\right)} - \underbrace{\kappa_c(\alpha \boldsymbol{x}_*)\, b_c}_{\Theta\left(1/\alpha^{\frac{3}{2}}\right)} . \end{aligned}$$

Thus, it is easy to see that $\lim_{\alpha \to \infty} z_{ic}(\alpha \boldsymbol{x}_*) = 0$. Hence we have

$$\begin{aligned} \lim_{\alpha \to \infty} \widetilde{p}(y_* = c \mid \alpha \boldsymbol{x}_*, \mathcal{D}) &= \lim_{\alpha \to \infty} \frac{1}{1 + \sum_{i \neq c}^{C} \exp(z_{ic}(\alpha \boldsymbol{x}_*))} \\ &= \frac{1}{1 + \sum_{i \neq c}^{C} \exp(0)} \\ &= \frac{1}{C}, \end{aligned}$$

as required. □

Appendix D Additional Details

D.1 Implementation

The kernel hyperparameters $(\sigma_l^2)_{l=0}^{L-1} =: \boldsymbol{\sigma}^2$ control the variance growth of the DSCS kernel since they come as multiplicative factors in (6). By increasing them, one can make the high confidence region more compact around the data. But if they are *too* large, the uncertainty will be high even in the data region, resulting in underconfidence.

Recent work has shown that outlier data can aid uncertainty calibration (Lee et al., 2018; Hendrycks et al., 2019, etc.). Following this line of work, we use the following objective, which intuitively balances high-confidence predictions on the data and low-confidence predictions on outliers. Let $p(y_* \mid \boldsymbol{x}_*, \mathcal{D}, \boldsymbol{\sigma}^2)$ be the predictive distribution under RGPR, H the entropy functional, $\mathcal{D}_{\text{val}}$ the validation set of $\mathcal{D}$, and $\mathcal{D}_{\text{out}}$ an outlier dataset. We define:[4]

$$\begin{aligned} L(\boldsymbol{\sigma}^2) := &\mathop{\mathbb{E}}_{\boldsymbol{x}_* \in \mathcal{D}_{\text{val}}} H\left(p(y_* \mid \boldsymbol{x}_*, \mathcal{D}; \boldsymbol{\sigma}^2)\right) \\ &- \mathop{\mathbb{E}}_{\boldsymbol{x} \in \mathcal{D}_{\text{out}}} H\left(p(y_* \mid \boldsymbol{x}_*, \mathcal{D}; \boldsymbol{\sigma}^2)\right), \end{aligned} \tag{10}$$

and optimize it via gradient descent—this process is quick since no backward pass through the network is required. We discuss the choice of $\mathcal{D}_{\text{out}}$ in Section 3.

Algorithm 1 provides a pseudocode of RGPR for classification predictions via MC-integration. The only overhead compared to the usual MC-integrated BNN prediction step are (marked in red) (i) a single additional forward-pass over $f_{\boldsymbol{\mu}}$, (ii) L evaluations of the DSCS kernel k, and (iii) sampling from a C-dimensional diagonal Gaussian. Their costs are negligible compared to the cost of obtaining the standard MC-prediction of f, which, in particular, requires multiple forward passes.

D.2 Review of Residual-GP Methods

The method of Blight & Ott (1975), henceforth called BNO, models the residual of polynomial regressions. That is, suppose $\boldsymbol{\phi} : \mathbb{R} \to \mathbb{R}^D$ is a polynomial basis function defined by $\boldsymbol{\phi}(x) := (1, x, x^2, \dots, x^{D-1})$, k is an arbitrary kernel, and $\boldsymbol{w} \in \mathbb{R}^D$ is a weight vector, BNO assumes

$$\widetilde{f}(x) := \boldsymbol{w}^\top \boldsymbol{\phi}(x) + \widehat{f}(x), \qquad \text{where } \widehat{f} \sim \mathcal{GP}(0, k).$$

Recently, this method has been extended to neural network. Qiu et al. (2020) apply the same idea—modeling residuals with GPs—to pre-trained networks, resulting in a method called RIO. Suppose

[4]One can also use the cross-entropy loss for the first term.

Algorithm 1 MC-prediction using RGPR. Differences from the standard procedure are in **red**.

```
Input:
    Pre-trained $L$-layer, ReLU BNN classifier $f$ with posterior $\mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$. Test point $\boldsymbol{x}_* \in \mathbb{R}^N$.
    Standardization function std. Hyperparameters $(\sigma_l^2)_{l=0}^{L-1}$ of $\widehat{f}$. Number of MC samples $S$.
1: $(\boldsymbol{h}_*^{(l)})_{l=1}^{L-1} = \texttt{forward}(f_{\boldsymbol{\mu}}, \boldsymbol{x}_*)$
2: $v_s(\boldsymbol{x}_*) = \sum_{l=0}^{L-1} k(\texttt{std}(\boldsymbol{h}_*^{(l)}), \texttt{std}(\boldsymbol{h}_*^{(l)}); \sigma_l^2)$
3: for $s = 1, \dots, S$ do
4:     $\boldsymbol{\theta}_s \sim \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$
5:     $\boldsymbol{f}_s(\boldsymbol{x}_*) = f(\boldsymbol{x}_*; \boldsymbol{\theta}_s)$
6:     $\widehat{\boldsymbol{f}}_s(\boldsymbol{x}_*) \sim \mathcal{N}(\boldsymbol{0}, v_s(\boldsymbol{x}_*)\boldsymbol{I})$
7:     $\widetilde{\boldsymbol{f}}_s(\boldsymbol{x}_*) = \boldsymbol{f}_s(\boldsymbol{x}_*) + \widehat{\boldsymbol{f}}_s(\boldsymbol{x}_*)$
8: end for
9: return $S^{-1} \sum_{s=1}^{S} \mathrm{softmax}(\widetilde{\boldsymbol{f}}_s(\boldsymbol{x}_*))$
```

Table 2: RGPRs compared to their respective base methods on the detection of far-away outliers. Values are average confidences. Error bars are standard errors over three prediction runs. For each dataset, the best value over each vanilla and RGPR-imbued method (e.g. KFL against KFL-RGPR) are in bold.

Methods	**CIFAR10**	**SVHN**
GP-DSCS	22.0±0.2	22.1±0.3
KFL	64.5±0.7	63.4±1.5
KFL-RGPR	**29.9**±0.3	**27.5**±0.0
SWAG	63.5±1.8	50.2±4.2
SWAG-RGPR	**29.3**±0.2	**27.5**±0.0
SVDKL	46.4±0.3	49.1±0.2
SVDKL-RGPR	**22.0**±0.1	**22.1**±0.1

that $f_{\boldsymbol{\mu}} : \mathbb{R}^N \to \mathbb{R}$ is a neural-network with a pre-trained, *point-estimated* parameters $\boldsymbol{\mu}$. Their method is defined by

$$\widetilde{f}(\boldsymbol{x}) := f_{\boldsymbol{\mu}}(\boldsymbol{x}) + \widehat{f}(\boldsymbol{x}), \qquad \text{where } \widehat{f} \sim \mathcal{GP}(0, k_{\mathrm{IO}}).$$

The kernel k_{IO} is a sum of RBF kernels applied on the dataset $\mathcal{D}$ (inputs) and the network's predictions over $\mathcal{D}$ (outputs), hence the name IO—input-output. As in the original Blight and Ott's method, RIO also focuses in modeling predictive residuals and requires GP posterior inference. Suppose that $m(\boldsymbol{x})$ and $v(\boldsymbol{x})$ is the a posteriori marginal mean and variance of the GP, respectively. Then, via standard computations, one can see that even though f is a point-estimated network, $\widetilde{f}$ is a random function, distributed *a posteriori* by

$$\widetilde{f}(\boldsymbol{x}) \sim \mathcal{N}\left(\widetilde{f}_{\boldsymbol{\mu}}(\boldsymbol{x}) + m(\boldsymbol{x}), v(\boldsymbol{x})\right).$$

Thus, BNO and RIO effectively add uncertainty to point-estimated networks.

The posterior inference of BNO and RIO can be computationally intensive, depending on the number of training examples M: The cost of exact posterior inference is in $\Theta(M^3)$. While it can be alleviated by approximate inference, such as via inducing point methods and stochastic optimizations, the posterior inference requirement can still be a hindrance for a practical adoption of BNO and RIO, especially on large problems.

Appendix E Additional Experiments

E.1 Asymptotic Regime

As a gold standard GP baseline, we compare against the method of Qiu et al. (2020) (with our DSCS kernel). We refer to this baseline simply as GP-DSCS. The base methods, which RGPR is implemented on, are the following recently-proposed BNNs: (i) Kronecker-factored Laplace (KFL, Ritter et al., 2018), (ii) stochastic weight averaging-Gaussian (SWAG, Maddox et al., 2019), and (iii) stochastic variational deep kernel learning (SVDKL, Wilson et al., 2016). All the kernel hyperparameters for RGPR are set to a constant value of 1×10^{-10} since we focus on the asymptotic regime. In all cases, MC-integral with 10 posterior samples is used for making predictions.

We construct a test dataset artificially by sampling 2000 uniform noises in $[0, 1]^N$ and scale them with a scalar $\alpha = 2000$. The goal is to achieve low confidence over these far-away points.

The results are presented in Table 2. We observe that the RGPR-augmented methods are significantly better than their respective base methods. In particular, their confidence estimates are significantly lower than those of the vanilla methods, becoming closer to the confidence of the gold-standard GP-DSCS baseline. This indicates that RGPR makes BNNs better calibrated in the asymptotic regime.

E.2 Training Details

For LeNet, we use Adam optimizer with an initial learning rate 1×10^{-3} while for ResNet, we use SGD with an initial learning rate 0.1 and momentum 0.9. In both cases, the optimization is carried out for 100 epochs using weight decay 5×10^{-4}. We also reduce the learning rate by a factor of 10 at epochs 50, 75, and 90. The test accuracies are 99%, 94%, 97%, and 76% for MNIST, CIFAR10, SVHN, and CIFAR100, respectively.

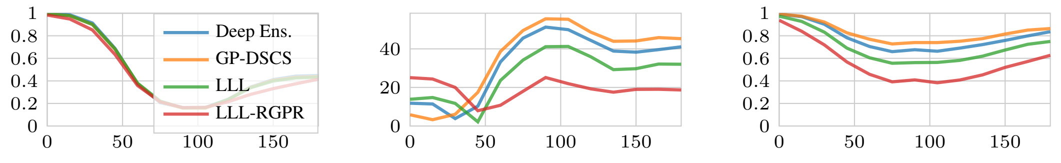
E.3 Non-Asymptotic Regime

Dataset shift We show more results for rotated-MNIST dataset in Figure 7, in terms of accuracy, expected calibration error (ECE), and mean confidence (MMC). RGPR makes LLL more calibrated in terms of ECE for rotation angles $>$ 45 degree. RGPR also makes LLL the most uncertain model for high rotation angles.

OOD detection We expand Table 1 in Table 3. For CIFAR10, SVHN, and CIFAR100, we test each model against FMNIST to measure the performance on grayscale OOD images. Finally, we also show the OOD detection performance via additional AUROC and area under precision-recall curve (AUPRC) metrics in Table 4.

Hyperparameter tuning We present the optimal hyperparameters $(\sigma_l^2)_{l=0}^{L-1}$ in Table 5. We observe that using higher representations of the data is beneficial, as indicated by non-trivial hyperparameter values on all layers across all networks and datasets.

Natural images for tuning We present OOD detection results via different $\mathcal{D}_{\text{our}}$ for tuning $\boldsymbol{\sigma}^2$, in Tables 6 and 7. Specifically, we use the ImageNet32x32 dataset (Chrabaszcz et al., 2017), which represents natural image datasets, and thus more sophisticated than the noise dataset used in the main text. Nevertheless, we observe that the performance is comparable to that of the noise dataset—justifying the choice we have made in the main text.



(a) Accuracy ↑ (b) ECE ↓ (c) MMC ↓

Figure 7: Rotated-MNIST results (averages of 10 prediction runs). x-axes are rotation angles. In **(a)**, all methods achieve similar accuracy.

Table 3: OOD data detection results in terms of MMC and FPR@95 metrics. All values are averages and standard errors over 10 prediction trials.

	MAP		**Deep Ens.**		**GP-DSCS**		**LLL**		**LLL-RGPR**	
Datasets	**MMC ↓**	**FPR ↓**	**MMC ↓**	**FPR ↓**	**MMC ↓**	**FPR ↓**	**MMC ↓**	**FPR ↓**	**MMC ↓**	**FPR ↓**
MNIST	99.2	-	99.1	-	99.2±0.0	-	97.4±0.0	-	93.7±0.0	-
EMNIST	78.1	24.5	74.1	**21.4**	77.6±0.0	24.7±0.0	62.7±0.0	23.3±0.0	**42.6**±0.0	30.5±0.1
KMNIST	73.1	14.3	63.1	5.6	72.2±0.0	13.2±0.0	52.8±0.0	6.3±0.0	**15.3**±0.0	**0.0**±0.0
FMNIST	79.8	26.8	71.7	11.3	79.1±0.0	25.5±0.0	64.6±0.0	19.0±0.1	**15.2**±0.0	**0.0**±0.0
GrayCIFAR10	85.7	3.6	72.7	**0.0**	85.2±0.0	3.5±0.0	61.1±0.0	0.5±0.0	**15.1**±0.0	**0.0**±0.0
UniformNoise	100.0	100.0	99.9	100.0	100.0±0.0	100.0±0.0	95.7±0.0	99.6±0.1	**15.1**±0.0	**0.0**±0.0
CIFAR10	97.0	-	95.6	-	96.9±0.0	-	93.4±0.0	-	91.9±0.0	-
SVHN	62.5	29.3	59.7	37.0	69.0±0.0	40.0±0.1	47.0±0.0	24.7±0.1	**45.1**±0.0	**24.0**±0.1
LSUN	74.5	52.7	65.6	50.3	76.6±0.0	55.2±0.1	58.5±0.1	44.2±0.3	**55.3**±0.0	**42.4**±0.3
CIFAR100	79.4	61.5	70.7	58.0	80.0±0.0	62.6±0.1	66.0±0.0	58.2±0.1	**62.9**±0.0	**56.8**±0.1
FMNIST3D	71.4	45.3	63.0	44.1	72.6±0.0	48.0±0.1	53.4±0.0	34.9±0.1	**47.4**±0.0	**27.5**±0.1
UniformNoise	64.6	25.7	74.0	86.2	75.5±0.1	54.7±0.5	39.0±0.1	2.8±0.2	**36.0**±0.1	**1.8**±0.1
SVHN	98.5	-	97.8	-	98.5±0.0	-	92.4±0.0	-	90.6±0.0	-
CIFAR10	70.4	18.3	57.2	**11.9**	70.9±0.0	19.8±0.0	41.7±0.0	14.9±0.1	**36.1**±0.0	13.2±0.1
LSUN	71.7	18.7	56.0	**10.0**	72.2±0.0	20.1±0.1	42.8±0.1	16.3±0.3	**32.3**±0.1	10.9±0.3
CIFAR100	71.3	20.4	57.6	**12.6**	71.8±0.0	22.2±0.0	43.2±0.0	17.8±0.1	**36.4**±0.0	14.5±0.1
FMNIST3D	72.5	21.9	61.9	20.0	72.8±0.0	23.0±0.0	45.3±0.0	21.8±0.1	**17.6**±0.0	**0.0**±0.0
UniformNoise	69.2	14.9	48.3	3.8	69.1±0.2	15.8±0.3	41.3±0.1	13.3±0.3	**26.5**±0.1	**2.7**±0.2
CIFAR100	81.3	-	80.2	-	82.2±0.0	-	74.4±0.0	-	66.4±0.0	-
SVHN	53.5	78.9	44.7	**65.5**	46.8±0.0	68.3±0.0	42.6±0.0	77.3±0.1	**37.0**±0.0	79.9±0.1
LSUN	50.7	74.7	47.1	76.0	53.5±0.0	76.9±0.1	39.6±0.0	**73.4**±0.4	**32.7**±0.1	77.3±0.4
CIFAR10	53.3	78.3	51.3	**76.9**	56.0±0.0	78.9±0.0	44.1±0.0	77.8±0.1	**37.0**±0.0	79.5±0.1
FMNIST3D	38.9	60.8	38.1	59.6	44.3±0.0	65.6±0.0	30.0±0.0	58.6±0.1	**17.5**±0.0	**39.0**±0.2
UniformNoise	29.4	55.5	45.4	95.7	31.6±0.1	50.3±0.3	22.0±0.0	46.9±0.6	**14.7**±0.0	**29.1**±0.5

Table 4: OOD data detection results in terms of AUROC and AUPRC metrics. All values are averages and standard errors over 10 prediction trials.

	MAP		Deep Ens.		GP-DSCS		LLL		LLL-RGPR	
Datasets	**ROC ↑**	**PRC ↑**	**ROC ↑**	**PRC ↑**	**ROC ↑**	**PRC ↑**	**ROC ↑**	**PRC ↑**	**ROC ↑**	**PRC ↑**
MNIST	-	-	-	-	-	-	-	-	-	-
EMNIST	95.0	89.6	**95.7**	**91.2**	94.8±0.0	89.0±0.0	94.2±0.0	86.8±0.0	93.3±0.0	86.4±0.0
KMNIST	96.0	93.0	98.3	97.6	96.4±0.0	93.7±0.0	98.4±0.0	98.3±0.0	**99.3**±0.0	**99.5**±0.0
FMNIST	92.2	85.8	96.6	94.0	92.7±0.0	86.5±0.0	96.8±0.0	96.9±0.0	**99.3**±0.0	**99.5**±0.0
GrayCIFAR10	98.0	98.5	99.0	99.4	98.0±0.0	98.6±0.0	98.5±0.0	99.0±0.0	**99.3**±0.0	**99.5**±0.0
UniformNoise	0.1	59.8	42.7	76.5	0.1±0.0	59.8±0.0	84.6±0.1	96.3±0.0	**99.3**±0.0	**99.9**±0.0
CIFAR10	-	-	-	-	-	-	-	-	-	-
SVHN	95.7	91.0	95.2	92.0	93.6±0.0	85.6±0.0	**96.3**±0.0	**92.2**±0.0	96.2±0.0	91.9±0.0
LSUN	91.8	99.6	92.8	**99.7**	90.7±0.0	99.6±0.0	92.7±0.0	**99.7**±0.0	**92.9**±0.0	**99.7**±0.0
CIFAR100	87.3	83.7	**90.1**	**89.5**	86.3±0.0	82.4±0.0	88.0±0.0	84.7±0.0	88.0±0.0	84.6±0.0
FMNIST3D	92.9	92.2	94.0	94.5	92.3±0.0	91.6±0.0	94.7±0.0	94.5±0.0	**95.9**±0.0	**96.0**±0.0
UniformNoise	96.7	99.3	92.7	98.4	94.3±0.0	98.7±0.0	98.8±0.0	99.7±0.0	**98.9**±0.0	**99.8**±0.0
SVHN	-	-	-	-	-	-	-	-	-	-
CIFAR10	95.4	97.0	97.5	98.9	95.0±0.0	96.7±0.0	97.3±0.0	98.9±0.0	**97.6**±0.0	**99.1**±0.0
LSUN	95.6	99.9	98.0	**100.0**	95.1±0.0	99.9±0.0	97.4±0.0	**100.0**±0.0	**98.3**±0.0	**100.0**±0.0
CIFAR100	94.5	96.4	97.3	98.7	94.1±0.0	96.1±0.0	96.8±0.0	98.7±0.0	**97.4**±0.0	**99.0**±0.0
FMNIST3D	94.2	96.4	96.5	98.5	94.1±0.0	96.4±0.0	96.0±0.0	98.2±0.0	**99.9**±0.0	**100.0**±0.0
UniformNoise	96.6	99.7	98.8	**99.9**	96.5±0.1	99.6±0.0	97.7±0.0	99.8±0.0	**99.1**±0.0	**99.9**±0.0
CIFAR100	-	-	-	-	-	-	-	-	-	-
SVHN	78.8	63.7	**84.6**	73.2	84.4±0.0	**73.3**±0.0	80.4±0.0	66.6±0.0	78.1±0.0	60.0±0.0
LSUN	81.1	99.1	**83.2**	**99.2**	80.3±0.0	99.1±0.0	82.5±0.0	**99.2**±0.0	81.9±0.0	**99.2**±0.0
CIFAR10	78.7	77.8	**80.1**	**79.6**	78.1±0.0	77.2±0.0	78.9±0.0	77.6±0.0	78.1±0.0	76.3±0.0
FMNIST3D	87.4	86.9	89.0	89.5	85.7±0.0	85.4±0.0	88.6±0.0	88.2±0.0	**93.3**±0.0	**93.1**±0.0
UniformNoise	93.4	98.5	86.2	96.9	93.3±0.0	98.5±0.0	94.3±0.0	98.7±0.0	**95.9**±0.0	**99.1**±0.0

Table 5: Optimal hyperparameter for each layer (or residual block for ResNet) on LLL.

Datasets	**Input**	**Layer 1**	**Layer 2**	**Layer 3**	**Layer 4**
MNIST	1.0567e-04	2.3090e-04	2.6643e-05	1.2558e-02	-
CIFAR10	1.0081e+01	1.0264e-03	6.6690e-04	1.8025e-03	1.5980e-03
SVHN	0.0034	0.0109	0.0016	0.0014	0.0078
CIFAR100	2.6449e+01	4.3422e-03	1.0548e-03, 3.6053e-03	7.6463e-03	

Table 6: OOD data detection results, where the ImageNet32x32 dataset is used for tuning, in terms of MMC and FPR@95 metrics . All values are averages and standard errors over 3 prediction trials.

	MAP		Deep Ens.		GP-DSCS		LLL		LLL-RGPR	
Datasets	**MMC ↓**	**FPR ↓**	**MMC ↓**	**FPR ↓**	**MMC ↓**	**FPR ↓**	**MMC ↓**	**FPR ↓**	**MMC ↓**	**FPR ↓**
MNIST	99.2	-	99.1	-	99.2±0.0	-	97.4±0.0	-	94.0±0.0	-
EMNIST	78.1	24.5	74.1	**21.4**	77.6±0.0	24.7±0.0	62.7±0.0	23.1±0.1	**43.8**±0.0	29.4±0.1
KMNIST	73.1	14.3	63.1	5.6	72.2±0.0	13.2±0.1	52.8±0.0	6.2±0.1	**15.3**±0.0	**0.0**±0.0
FMNIST	79.8	26.8	71.7	11.3	79.1±0.0	25.6±0.1	64.5±0.0	18.6±0.3	**15.3**±0.0	**0.0**±0.0
GrayCIFAR10	85.7	3.6	72.7	**0.0**	85.2±0.0	3.6±0.0	61.1±0.0	0.4±0.0	**15.1**±0.0	**0.0**±0.0
UniformNoise	100.0	100.0	99.9	99.9	100.0±0.0	100.0±0.0	95.6±0.0	99.3±0.1	**15.1**±0.0	**0.0**±0.0
CIFAR10	97.0	-	95.6	-	96.9±0.0	-	93.4±0.0	-	85.5±0.0	-
SVHN	62.5	29.3	59.7	37.0	69.0±0.0	40.1±0.2	47.0±0.0	**24.8**±0.2	**45.6**±0.0	46.3±0.4
LSUN	74.5	52.7	65.6	50.3	76.6±0.0	55.6±0.3	58.6±0.1	45.1±0.4	**41.2**±0.1	**40.9**±0.5
CIFAR100	79.4	61.5	70.7	**58.0**	80.0±0.0	62.7±0.2	66.0±0.0	58.3±0.2	**55.6**±0.0	67.1±0.3
FMNIST3D	71.4	45.3	63.0	44.1	72.6±0.0	48.0±0.2	53.4±0.0	35.1±0.3	**32.3**±0.0	**18.0**±0.4
UniformNoise	64.7	25.9	74.2	86.4	75.5±0.3	55.8±0.9	39.1±0.2	2.9±0.3	**17.4**±0.0	**0.0**±0.0
SVHN	98.5	-	97.8	-	98.5±0.0	-	92.4±0.0	-	88.2±0.0	-
CIFAR10	70.4	18.3	57.2	11.9	70.9±0.0	19.9±0.1	41.7±0.0	14.8±0.1	**29.2**±0.0	**10.6**±0.1
LSUN	71.7	18.7	56.0	10.0	72.2±0.0	19.9±0.1	42.7±0.1	16.0±0.2	**23.3**±0.1	**4.2**±0.8
CIFAR100	71.3	20.4	57.6	12.6	71.8±0.0	22.2±0.0	43.2±0.0	17.5±0.1	**29.3**±0.0	**11.4**±0.1
FMNIST3D	72.5	21.9	61.9	20.0	72.8±0.0	22.9±0.0	45.3±0.0	21.5±0.1	**15.4**±0.0	**0.0**±0.0
UniformNoise	68.8	14.8	48.7	4.4	68.6±0.2	15.2±0.1	41.1±0.2	13.2±0.1	**18.4**±0.1	**0.2**±0.0
CIFAR100	81.3	-	80.2	-	82.2±0.0	-	74.4±0.0	-	65.2±0.0	-
SVHN	53.5	78.9	44.7	**65.5**	46.8±0.0	68.3±0.1	42.6±0.0	77.7±0.2	**40.5**±0.0	87.7±0.1
LSUN	50.7	74.7	47.1	76.0	53.5±0.0	77.0±0.3	39.8±0.1	74.1±0.3	**26.4**±0.1	**68.6**±0.6
CIFAR10	53.3	78.3	51.3	**76.9**	56.0±0.0	78.9±0.1	44.1±0.0	78.2±0.3	**35.2**±0.0	81.0±0.1
FMNIST3D	38.9	60.8	38.1	59.6	44.3±0.0	65.6±0.0	30.0±0.0	59.1±0.4	**17.5**±0.0	**43.6**±0.2
UniformNoise	29.4	55.9	45.3	95.0	31.5±0.1	50.4±0.6	22.0±0.1	48.2±0.9	**4.7**±0.0	**0.0**±0.0

Table 7: OOD data detection results, where the ImageNet32x32 dataset is used for tuning, in terms of AUROC and AUPRC metrics. All values are averages and standard errors over 3 prediction trials.

	MAP		**Deep Ens.**		**GP-DSCS**		**LLL**		**LLL-RGPR**	
Datasets	**ROC ↑**	**PRC ↑**	**ROC ↑**	**PRC ↑**	**ROC ↑**	**PRC ↑**	**ROC ↑**	**PRC ↑**	**ROC ↑**	**PRC ↑**
MNIST	-	-	-	-	-	-	-	-	-	-
EMNIST	95.0	89.6	**95.7**	**91.2**	94.8±0.0	89.0±0.0	94.2±0.0	86.8±0.0	93.3±0.0	86.4±0.0
KMNIST	96.0	93.0	98.3	97.6	96.4±0.0	93.7±0.0	98.4±0.0	98.3±0.0	**99.4**±0.0	**99.5**±0.0
FMNIST	92.2	85.8	96.6	94.0	92.7±0.0	86.4±0.0	96.8±0.0	96.9±0.0	**99.4**±0.0	**99.5**±0.0
GrayCIFAR10	98.0	98.5	99.0	99.4	98.0±0.0	98.6±0.0	98.5±0.0	99.0±0.0	**99.4**±0.0	**99.6**±0.0
UniformNoise	0.1	59.8	42.7	76.6	0.1±0.0	59.8±0.0	84.6±0.1	96.3±0.0	**99.4**±0.0	**99.9**±0.0
CIFAR10	-	-	-	-	-	-	-	-	-	-
SVHN	95.7	91.0	95.2	92.0	93.6±0.0	85.6±0.0	**96.3**±0.0	**92.2**±0.0	91.6±0.0	82.9±0.0
LSUN	91.8	99.6	92.8	99.7	90.7±0.0	99.6±0.0	92.7±0.0	99.7±0.0	**93.8**±0.0	**99.8**±0.0
CIFAR100	87.3	83.7	**90.1**	**89.5**	86.3±0.0	82.4±0.0	88.0±0.0	84.7±0.0	85.1±0.0	81.9±0.0
FMNIST3D	92.9	92.2	94.0	94.5	92.3±0.0	91.6±0.0	94.7±0.0	94.5±0.0	**96.7**±0.0	**97.2**±0.0
UniformNoise	96.7	99.3	92.7	98.4	94.3±0.0	98.7±0.0	98.8±0.0	99.7±0.0	**99.4**±0.0	**99.9**±0.0
SVHN	-	-	-	-	-	-	-	-	-	-
CIFAR10	95.4	97.0	97.5	98.9	95.0±0.0	96.7±0.0	97.3±0.0	98.9±0.0	**97.9**±0.0	**99.2**±0.0
LSUN	95.6	99.9	98.0	**100.0**	95.1±0.0	99.9±0.0	97.4±0.0	**100.0**±0.0	**98.9**±0.0	**100.0**±0.0
CIFAR100	94.5	96.4	97.3	98.7	94.1±0.0	96.1±0.0	96.9±0.0	98.7±0.0	**97.8**±0.0	**99.1**±0.0
FMNIST3D	94.2	96.4	96.5	98.5	94.1±0.0	96.4±0.0	96.0±0.0	98.2±0.0	**99.9**±0.0	**100.0**±0.0
UniformNoise	96.7	99.7	98.8	99.9	96.6±0.0	99.7±0.0	97.7±0.0	99.8±0.0	**99.6**±0.0	**100.0**±0.0
CIFAR100	-	-	-	-	-	-	-	-	-	-
SVHN	78.8	63.7	**84.6**	73.2	84.4±0.0	**73.3**±0.0	80.3±0.0	66.6±0.0	73.9±0.0	54.4±0.0
LSUN	81.1	99.1	83.2	99.2	80.3±0.0	99.1±0.0	82.4±0.1	99.2±0.0	**85.7**±0.0	**99.4**±0.0
CIFAR10	78.7	77.8	**80.1**	**79.6**	78.1±0.0	77.2±0.0	78.9±0.0	77.6±0.0	78.4±0.0	77.6±0.0
FMNIST3D	87.4	86.9	89.0	89.5	85.7±0.0	85.4±0.0	88.6±0.0	88.1±0.0	**92.4**±0.0	**92.1**±0.0
UniformNoise	93.4	98.5	86.3	96.9	93.3±0.0	98.5±0.0	94.3±0.0	98.7±0.0	**99.9**±0.0	**100.0**±0.0