Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks

Agustinus Kristiadi [1] **Matthias Hein** [1] **Philipp Hennig** [1 2]

Abstract

The point estimates of ReLU classification networks—arguably the most widely used neural network architecture—have been shown to yield arbitrarily high confidence far away from the training data. This architecture, in conjunction with a maximum a posteriori estimation scheme, is thus not calibrated nor robust. Approximate Bayesian inference has been empirically demonstrated to improve predictive uncertainty in neural networks, although the theoretical analysis of such Bayesian approximations is limited. We theoretically analyze approximate Gaussian distributions on the weights of ReLU networks and show that they fix the overconfidence problem. Furthermore, we show that even a simplistic, thus cheap, Bayesian approximation, also fixes these issues. This indicates that a sufficient condition for a calibrated uncertainty on a ReLU network is "to be a bit Bayesian". These theoretical results validate the usage of last-layer Bayesian approximation and motivate a range of a fidelity-cost trade-off. We further validate these findings empirically via various standard experiments using common deep ReLU networks and Laplace approximations.

1. Introduction

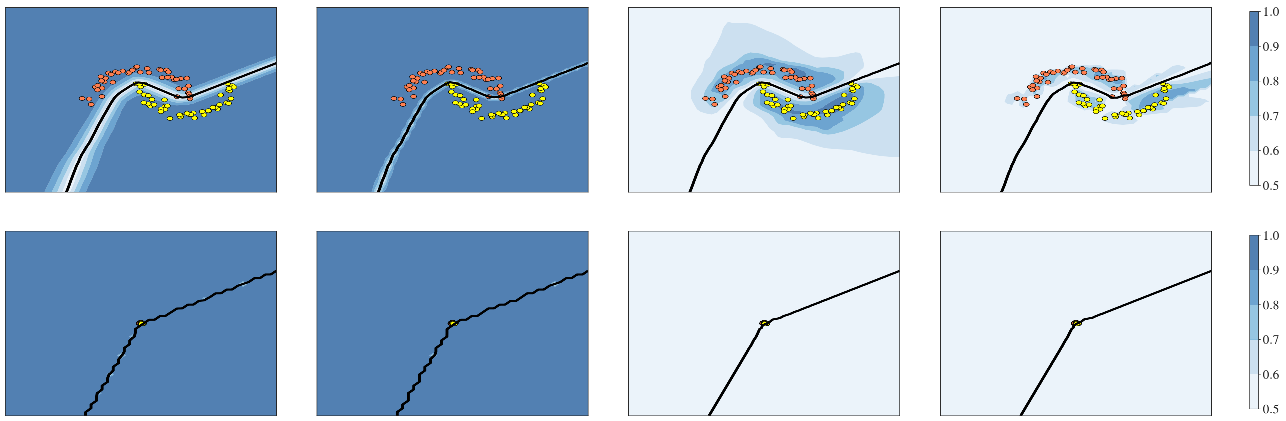
As neural networks have been successfully applied in ever more domains, including safety-critical ones, the robustness and uncertainty quantification of their predictions have moved into focus, subsumed under the notion of AI safety (Amodei et al., 2016). A principal goal of uncertainty quantification is that learning machines and neural networks in particular, should assign low confidence to test cases not explained well by the training data or prior information (Gal, 2016). The most obvious cases are test points that lie "far away" from the training data. Many methods to achieve this goal have been proposed, both Bayesian (e.g. Blundell et al., 2015; Louizos & Welling, 2017; Zhang et al., 2018) and non-Bayesian (e.g. Lakshminarayanan et al., 2017; Liang et al., 2018; Hein et al., 2019).

ReLU networks are currently among the most widely used neural architectures. This class comprises any network that can be written as a composition of linear layers (including fully-connected, convolutional, and residual layers) and a ReLU activation function. But, while ReLU networks often achieve high accuracy, the *uncertainty* of their predictions has been shown to be miscalibrated (Guo et al., 2017). Indeed, Hein et al. (2019) demonstrated that ReLU networks are *always* overconfident "far away from the data": scaling a training point $\mathbf{x}$ (a vector in a Euclidean input space) with a scalar δ yields predictions of arbitrarily high confidence in the limit $\delta \to \infty$. This means ReLU networks are susceptible to out-of-distribution (OOD) examples. Meanwhile, probabilistic methods (in particular Bayesian methods) have long been known empirically to improve predictive uncertainty estimates. MacKay (1992a) demonstrated experimentally that the predictive uncertainty of Bayesian neural networks will naturally be high in regions not covered by training data. Although the theoretical analysis is still lacking, results like this raise the hope that the overconfidence problem of ReLU networks, too, might be mitigated by the use of probabilistic and Bayesian methods.

This paper offers a theoretical analysis of the binary classification case of ReLU networks with a logistic output layer. We show that equipping such networks with a Gaussian approximate distribution over the weights mitigates the aforementioned theoretical problem, in the sense that the predictive confidence far away from the training data approaches a known limit, bounded away from one, whose value is controlled by the covariance. In the case of Laplace approximations (MacKay, 1992b; Ritter et al., 2018), this treatment in conjunction with the probit approximation (Spiegelhalter & Lauritzen, 1990; MacKay, 1992a) does not change the decision boundary of the trained network, so it has no negative effect on the predictive performance (cf. Figure 1). Furthermore, we show that a sufficient condition for this desirable property to hold is to apply a Gaussian approximation *only*

[1] University of Tübingen [2] MPI for Intelligent Systems, Tübingen. Correspondence to: Agustinus Kristiadi <agustinus.kristiadi@uni-tuebingen.de>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).



(a) MAP (b) Temp. scaling (c) Bayesian (last-layer) (d) Bayesian (all-layer)

Figure 1. Binary classification on a toy dataset using a MAP estimate, temperature scaling, and both last-layer and all-layer Gaussian approximations over the weights which are obtained via Laplace approximations. Background color and black line represent confidence and decision boundary, respectively. Bottom row shows a zoomed-out view of the top row. The Bayesian approximations—even in the last-layer case—give desirable uncertainty estimates: confident close to the training data and uncertain otherwise. MAP and temperature scaling yield overconfident predictions. The optimal temperature is picked as in Guo et al. (2017).

to the last layer of a ReLU network. This motivates the commonly used approximation scheme where an L-layer network is decomposed into a fixed feature map composed by the first $L-1$ layers and a Bayesian linear classifier (Gelman et al., 2008; Wilson et al., 2016a; Riquelme et al., 2018; Ober & Rasmussen, 2019; Brosse et al., 2020, etc.). This particular result implies that just being "a bit" Bayesian—at low cost overhead—already gives desirable benefits.

We empirically validate our results through various Laplace approximations on common deep ReLU networks. Furthermore, while our theoretical analysis is focused on the binary classification case, we also experimentally show that these Bayesian approaches yield good performance in the multi-class classification setting, suggesting that our analysis may carry over to this case.

To summarize, our contributions are three-fold:

(i) we provide theoretical analysis on why ReLU networks equipped with Gaussian distributions over the weights mitigate the overconfidence problem in the binary classification setting,

(ii) we show that a sufficient condition for having this property is to be "a bit" Bayesian: employing a last-layer Gaussian approximation—in particular a Bayesian one, and

(iii) we validate our theoretical findings via a series of comprehensive experiments involving commonly-used deep ReLU networks and Laplace approximations in both binary and multi-class cases.

Section 2 begins with definitions, assumptions, and the problem statement, then develops the main theoretical results. Proofs are available in Appendix A. We discuss related work in Section 3, while empirical results are shown in Section 4.

2. Analysis

2.1. Preliminaries

Definitions We call a function $f : \mathbb{R}^n \to \mathbb{R}^k$ piecewise affine if there exists a finite set of polytopes $\{Q_r\}_{r=1}^R$, referred to as the *linear regions* of f, such that $\cup_{r=1}^R Q_r = \mathbb{R}^n$ and $f|_{Q_r}$ is an affine function for every Q_r. ReLU networks are networks that result in piecewise affine classifier functions (Arora et al., 2018), which include networks with fully-connected, convolutional, and residual layers where just ReLU or leaky-ReLU are used as activation functions and max or average pooling are used in convolution layers. Let $\mathcal{D} := \{\mathbf{x}_i \in \mathbb{R}^n, t_i\}_{i=1}^m$ be a dataset, where the targets are $t_i \in \{0,1\}$ or $t_i \in \{1,\dots,k\}$ for the binary and multi-class case, respectively. Let $\phi : \mathbb{R}^n \to \mathbb{R}^d$ be an arbitrary fixed feature map and write $\boldsymbol{\phi} := \phi(\mathbf{x})$ for a given $\mathbf{x}$. We define the logistic (*sigmoid*) function as $\sigma(z) := 1/(1+\exp(-z))$ for $z \in \mathbb{R}$ and the softmax function as $\mathrm{softmax}(\mathbf{z}, i) := \exp(z_i)/\sum_j \exp(z_j)$ for $\mathbf{z} \in \mathbb{R}^k$ and $i \in \{1,\dots,k\}$. Given a neural network $f_{\boldsymbol{\theta}}$, we consider the distribution $p(\boldsymbol{\theta}|\mathcal{D})$ over its parameters. Note that even though we use the notation $p(\boldsymbol{\theta}|\mathcal{D})$, we do *not* require this distribution to be a posterior distribution in the Bayesian sense. The *predictive* distribution for the binary case is

$$p(y=1|\mathbf{x},\mathcal{D}) = \int \sigma(f_{\boldsymbol{\theta}}(\mathbf{x}))\, p(\boldsymbol{\theta}|\mathcal{D})\, d\boldsymbol{\theta}\,, \tag{1}$$

and for the multi-class case

$$p(y = i|\mathbf{x}, \mathcal{D}) = \int \operatorname{softmax}(f_{\boldsymbol{\theta}}(\mathbf{x}), i)\, p(\boldsymbol{\theta}|\mathcal{D})\, d\boldsymbol{\theta} \,. \tag{2}$$

The functions $\lambda_i(\cdot)$, $\lambda_{\text{max}}(\cdot)$, and $\lambda_{\text{min}}(\cdot)$ return the ith, maximum, and minimum eigenvalue (which are assumed to exist) of their matrix argument, respectively.[1] Similarly for the function $s_i(\cdot)$, $s_{\text{max}}(\cdot)$, and $s_{\text{min}}(\cdot)$ which return singular values instead. Finally, we assume that $\|\cdot\|$ is the ℓ^2 norm.

Problem statement The following theorem from Hein et al. (2019) shows that ReLU networks exhibit arbitrarily high confidence far away from the training data: If a point $\mathbf{x} \in \mathbb{R}^n$ is scaled by a sufficiently large scalar $\delta > 0$, the input $\delta\mathbf{x}$ attains arbitrarily high confidence.

Theorem 2.1 (Hein et al., 2019)**.** *Let* $\mathbb{R}^d = \cup_{r=1}^R Q_r$ *and* $f|_{Q_r}(\mathbf{x}) = \mathbf{U}_r\mathbf{x} + \mathbf{c}_r$ *be the piecewise affine representation of the output of a ReLU network on* Q_r. *Suppose that* $\mathbf{U}_r$ *does not contain identical rows for all* $r = 1, \dots, R$, *then for almost any* $\mathbf{x} \in \mathbb{R}^n$ *and any* $\epsilon > 0$, *there exists a* $\delta > 0$ *and a class* $i \in \{1, \dots, k\}$ *such that it holds* $\operatorname{softmax}(f(\delta\mathbf{x}), i) \geq 1 - \epsilon$. *Moreover,* $\lim_{\delta\to\infty} \operatorname{softmax}(f(\delta\mathbf{x}), i) = 1$. □

It is standard to treat neural networks as probabilistic models of the conditional distribution $p(y|\mathbf{x}, \boldsymbol{\theta})$ over the prediction y. In this case, we define the confidence of any input point $\mathbf{x}$ as the maximum predictive probability, which in the case of a binary problem, can be written as $\max_{i\in\{0,1\}} p(y = i|\mathbf{x}, \boldsymbol{\theta}) = \sigma(|f_{\boldsymbol{\theta}}(\mathbf{x})|)$. Standard training involves assigning a maximum a posteriori (MAP) estimate $\boldsymbol{\theta}_{\text{MAP}}$ to the weights, ignoring potential uncertainty on $\boldsymbol{\theta}$. We will show that this lack of uncertainty is the primary cause of the overconfidence discussed by Hein et al. (2019) and argue that it can be mitigated by considering the marginalized prediction in (1) instead.

Even for a linear classifier parametrized by a single weight matrix $\boldsymbol{\theta} = \mathbf{w}$, there is generally no analytic solution for (1). But, good approximations exist when the distribution over the weights is a Gaussian $p(\mathbf{w}|\mathcal{D}) \approx \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. One such approximation (Spiegelhalter & Lauritzen, 1990; MacKay, 1992a) is constructed by scaling the input of the *probit* function[2] Φ by a constant $\lambda = \sqrt{\pi/8}$. Using this approximation and the Gaussian assumption, if we let $a := \mathbf{w}^\top\boldsymbol{\phi}$, we get

$$\begin{aligned} p(y = 1|\mathbf{x}, \mathcal{D}) &\approx \int \Phi(\sqrt{\pi/8}\, a)\, \mathcal{N}(a|\boldsymbol{\mu}^\top\boldsymbol{\phi}, \boldsymbol{\phi}^\top\boldsymbol{\Sigma}\boldsymbol{\phi})\, da \\ &= \Phi\left(\frac{\boldsymbol{\mu}^\top\boldsymbol{\phi}}{\sqrt{8/\pi + \boldsymbol{\phi}^\top\boldsymbol{\Sigma}\boldsymbol{\phi}}}\right) \approx \sigma\left(z(\mathbf{x})\right) \,, \end{aligned} \tag{3}$$

where the last step uses the approximation $\Phi(\sqrt{\pi/8}\, x) \approx \sigma(x)$ a second time, with

$$z(\mathbf{x}) := \frac{\boldsymbol{\mu}^\top\boldsymbol{\phi}}{\sqrt{1 + \pi/8\, \boldsymbol{\phi}^\top\boldsymbol{\Sigma}\boldsymbol{\phi}}} \,. \tag{4}$$

In the case of $\boldsymbol{\mu} = \mathbf{w}_{\text{MAP}}$, Equation (3) can be seen as the "softened" version of the MAP prediction of the classifier, using the covariance of the Gaussian. The confidence in this case is $\max_{i\in\{0,1\}} p(y = i|\mathbf{x}, \mathcal{D}) = \sigma(|z(\mathbf{x})|)$.

We can generalize the previous insight to the case where the parameters of the feature map ϕ are also approximated by a Gaussian. Let $\boldsymbol{\theta} \in \mathbb{R}^p$ be the parameter vector of a NN $f_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}$ with a given Gaussian approximation $p(\boldsymbol{\theta}|\mathcal{D}) \approx \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Let $\mathbf{x} \in \mathbb{R}^n$ be an arbitrary input point. Letting $\mathbf{d} := \nabla f_{\boldsymbol{\theta}}(\mathbf{x})|_{\boldsymbol{\mu}}$, we do a first-order Taylor expansion of $f_{\boldsymbol{\theta}}$ at $\boldsymbol{\mu}$ (MacKay, 1995): $f_{\boldsymbol{\theta}}(\mathbf{x}) \approx f_{\boldsymbol{\mu}}(\mathbf{x}) + \mathbf{d}^\top(\boldsymbol{\theta} - \boldsymbol{\mu})$. This implies that the distribution over $f_{\boldsymbol{\theta}}(\mathbf{x})$ is given by $p(f_{\boldsymbol{\theta}}(\mathbf{x})|\mathbf{x}, \mathcal{D}) \approx \mathcal{N}(f_{\boldsymbol{\theta}}(\mathbf{x})|f_{\boldsymbol{\mu}}(\mathbf{x}), \mathbf{d}^\top\boldsymbol{\Sigma}\mathbf{d})$. Therefore, we have

$$z(\mathbf{x}) := \frac{f_{\boldsymbol{\mu}}(\mathbf{x})}{\sqrt{1 + \pi/8\, \mathbf{d}^\top\boldsymbol{\Sigma}\mathbf{d}}} \,. \tag{5}$$

It is easy to see that (4) is indeed a special case of (5).

As the first notable property of this approximation, we show that, in contrast to some other methods for uncertainty quantification (e.g. Monte Carlo dropout, Gal & Ghahramani, 2016) it preserves the decision boundary induced by the MAP estimate.

Proposition 2.2 (Invariance property)**.** *Let* $f_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}$ *be a binary classifier network parametrized by* $\boldsymbol{\theta}$ *and let* $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ *be the distribution over* $\boldsymbol{\theta}$. *Then for any* $\mathbf{x} \in \mathbb{R}^n$, *we have* $\sigma(f_{\boldsymbol{\mu}}(\mathbf{x})) = 0.5$ *if and only if* $\sigma(z(\mathbf{x})) = 0.5$.

This property is useful in practice, particularly whenever $\boldsymbol{\mu} = \boldsymbol{\theta}_{\text{MAP}}$, since it guarantees that employing a Gaussian approximation on top of a MAP-trained network will not reduce the original classification accuracy. Virtually all state-of-the-art models in deep learning are trained via MAP estimation and sacrificing the classification performance that makes them attractive in the first place would be a waste.

2.2. Main Results

As our central theoretical contribution, we show that for any $\mathbf{x} \in \mathbb{R}^n$, as $\delta \to \infty$, the value of $|z(\delta\mathbf{x})|$ in (5) goes to a quantity that only depends on the mean and covariance of the Gaussian over the weights. Moreover, this property also holds in the finite asymptotic regime, far enough from the training data. This result implies that one can drive the confidence closer to the uniform (one-half) far away from the training points by shifting $|z(\mathbf{x})|$ closer to zero by controlling the Gaussian. We formalize this result in the following theorem. The situation is illustrated in Figure 2.

[1] We assume they are sorted in a descending order.

[2] The probit function Φ is another sigmoid, the distribution function (CDF) of the standard Gaussian.

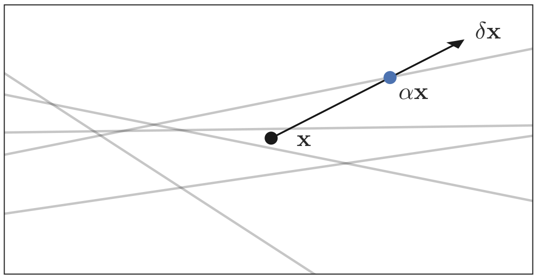


Figure 2. The situation in Theorems 2.3 and 2.4. The intersections of gray lines are the linear regions.

Theorem 2.3 (All-layer approximation)**.** *Let $f_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}$ be a binary ReLU classification network parametrized by $\boldsymbol{\theta} \in \mathbb{R}^p$ with $p \geq n$, and let $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be the Gaussian approximation over the parameters. Then for any input $\mathbf{x} \in \mathbb{R}^n$,*

$$\lim_{\delta\to\infty} \sigma(|z(\delta\mathbf{x})|) \leq \sigma\left(\frac{\|\mathbf{u}\|}{s_{\text{min}}(\mathbf{J})\sqrt{\pi/8\,\lambda_{\text{min}}(\boldsymbol{\Sigma})}}\right), \tag{6}$$

where $\mathbf{u} \in \mathbb{R}^n$ is a vector depending only on $\boldsymbol{\mu}$ and the $n \times p$ matrix $\mathbf{J} := \frac{\partial \mathbf{u}}{\partial \boldsymbol{\theta}}\big|_{\boldsymbol{\mu}}$ is the Jacobian of $\mathbf{u}$ w.r.t. $\boldsymbol{\theta}$ at $\boldsymbol{\mu}$. Moreover, if $f_{\boldsymbol{\theta}}$ has no bias parameters, then there exists $\alpha > 0$ such that for any $\delta \geq \alpha$, we have that

$$\sigma(|z(\delta\mathbf{x})|) \leq \lim_{\delta\to\infty} \sigma(|z(\delta\mathbf{x})|) \,.$$

The following question is practically interesting: Do we have to construct a probabilistic (Gaussian) uncertainty to the whole ReLU network for the previous property (Theorem 2.3) to hold? Surprisingly, the answer is no. The following theorem establishes that a guarantee similar to Theorem 2.3, is feasible even if *only the last layer's weights* are assigned a Gaussian distribution. This amounts to a form of Bayesian logistic regression, where the features are provided by the ReLU network.

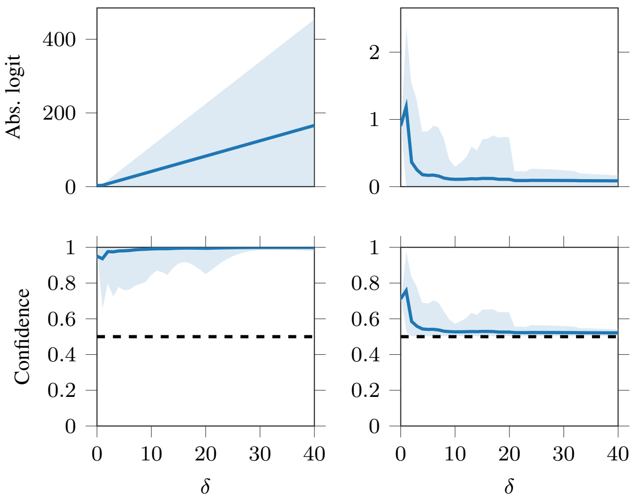
Theorem 2.4 (Last-layer approximation)**.** *Let $g : \mathbb{R}^d \to \mathbb{R}$ be a binary linear classifier defined by $g(\phi(\mathbf{x})) := \mathbf{w}^\top \phi(\mathbf{x})$ where $\phi : \mathbb{R}^n \to \mathbb{R}^d$ is a fixed ReLU network and let $\mathcal{N}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be the Gaussian approximation over the last-layer's weights. Then for any input $\mathbf{x} \in \mathbb{R}^n$,*

$$\lim_{\delta\to\infty} \sigma(|z(\delta\mathbf{x})|) \leq \sigma\left(\frac{\|\boldsymbol{\mu}\|}{\sqrt{\pi/8\,\lambda_{\text{min}}(\boldsymbol{\Sigma})}}\right). \tag{7}$$

Moreover, if ϕ has no bias parameters, then there exists $\alpha > 0$ such that for any $\delta \geq \alpha$, we have that

$$\sigma(|z(\delta\mathbf{x})|) \leq \lim_{\delta\to\infty} \sigma(|z(\delta\mathbf{x})|) \,.$$

We show, using the same toy dataset and Gaussian-based last-layer Bayesian method as in Figure 1, an illustration of the previous results in Figure 3. Confirming the findings, for each input $\mathbf{x}$, the Gaussian approximation drives $|z(\delta\mathbf{x})|$ to a constant for sufficiently large δ. Note that on true data points ($\delta = 1$), the confidences remain high and the convergence occurs at some finite δ.



(a) MAP

(b) Bayesian

Figure 3. Absolute logit—$|f_{\boldsymbol{\theta}_{\text{MAP}}}(\delta\mathbf{x})|$ for MAP and $|z(\delta\mathbf{x})|$ for Bayesian—and confidence of the toy dataset in Figure 1 as functions of δ. Each plot shows the mean and ± 3 standard deviation over the test set.

Taken together, the results above formally validate the usage of the common Gaussian approximations of the weights distribution, both in Bayesian (MacKay, 1992b; Graves, 2011; Blundell et al., 2015, etc.) or non-Bayesian (Franchi et al., 2019; Lu et al., 2020, etc.) fashions, on ReLU networks for mitigating overconfidence problems. Furthermore, Theorem 2.4 shows that a full-blown Gaussian approximation or Bayesian treatment (i.e. on all layers of a NN) is not required to achieve control over the confidence far away from the training data. Put simply, even being "just a bit Bayesian" is enough to overcome at least asymptotic overconfidence.

We will show in the experiments (Section 4.3) that the same Bayesian treatment also mitigates asymptotic confidence in the multi-class case. However, extending the theoretical analysis to this case is not straightforward, even with analytic approximations such as those by Gibbs (1998) and Wu et al. (2019).

2.3. Laplace Approximations

The results in the previous section imply that the asymptotic confidence of a Gaussian-approximated binary ReLU classifier—either via a full or last-layer approximation—can be driven closer to uniform by controlling the covariance. In this section, we analyze the case when a Bayesian method in the form of a Laplace approximation is employed for obtaining the Gaussian. Although Laplace approximations are currently less popular than variational Bayes (VB), they have useful practical benefits: (i) they can be applied to any pre-trained network, (ii) whenever the approximation (5) can be employed, Proposition 2.2 holds, and (iii) no re-training is needed. Indeed, Laplace approximations can be attractive to practitioners who already have a working MAP-trained network, but want to enhance its uncertainty estimates further without decreasing performance.

The principle of Laplace approximations is as follows. Let $p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\boldsymbol{\theta}) \prod_{\mathbf{x},t\in\mathcal{D}} p(y=t|\mathbf{x},\boldsymbol{\theta})$ be the posterior of a network $f_{\boldsymbol{\theta}}$. Then we can obtain a Gaussian approximation $p(\boldsymbol{\theta}|\mathcal{D}) \approx \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu},\boldsymbol{\Sigma})$ of the posterior by setting $\boldsymbol{\mu} = \boldsymbol{\theta}_{\text{MAP}}$ and $\boldsymbol{\Sigma} := (-\nabla^2 \log p(\boldsymbol{\theta}|\mathcal{D})|_{\boldsymbol{\theta}_{\text{MAP}}})^{-1}$, the inverse Hessian of the negative log-posterior at the mode. In the binary classification case, the likelihood $p(y|\mathbf{x},\mathbf{w})$ is assumed to be a Bernoulli distribution $\mathcal{B}(\sigma(f_{\boldsymbol{\theta}}(\mathbf{x})))$. The prior $p(\boldsymbol{\theta})$ is assumed to be an isotropic Gaussian $\mathcal{N}(\boldsymbol{\theta}|\mathbf{0},\sigma_0^2\mathbf{I})$.

While the prior variance σ_0^2 is tied to the MAP estimation (it can be derived from the weight decay), it is often treated as a separate hyperparameter and tuned after training (Ritter et al., 2018). This treatment is useful in the case when one has only a pre-trained network and not the original training hyperparameters. Under this situation, in the following proposition, we analyze the effect of σ_0^2 on the asymptotic confidence presented by Theorem 2.3. The statement for the last-layer case is analogous and presented in Appendix A.

Proposition 2.5 (All-layer Laplace)**.** *Let $f_{\boldsymbol{\theta}}$ be a binary ReLU classification network modeling a Bernoulli distribution $p(y|\mathbf{x},\boldsymbol{\theta}) = \mathcal{B}(\sigma(f_{\boldsymbol{\theta}}(\mathbf{x})))$ with parameter $\boldsymbol{\theta} \in \mathbb{R}^p$. Let $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu},\boldsymbol{\Sigma})$ be the posterior obtained via a Laplace approximation with prior $\mathcal{N}(\boldsymbol{\theta}|\mathbf{0},\sigma_0^2\mathbf{I})$, $\mathbf{H}$ be the Hessian of the negative log-likelihood at $\boldsymbol{\mu}$, and $\mathbf{J}$ be the Jacobian as in Theorem 2.3. Then for any input $\mathbf{x} \in \mathbb{R}^n$, the confidence $\sigma(|z(\mathbf{x})|)$ is a decreasing function of σ_0^2 with limits*

$$\lim_{\sigma_0^2\to\infty} \sigma(|z(\mathbf{x})|) \leq \sigma\left(\frac{|f_{\boldsymbol{\mu}}(\mathbf{x})|}{1+\sqrt{\pi/8\,\lambda_{\max}(\mathbf{H})\|\mathbf{J}\mathbf{x}\|^2}}\right)$$

$$\lim_{\sigma_0^2\to 0} \sigma(|z(\mathbf{x})|) = \sigma(|f_{\boldsymbol{\mu}}(\mathbf{x})|)\,.$$

The result above shows that the "far-away" confidence decreases (up to some limit) as the prior variance increases. Meanwhile, we recover the far-away confidence induced by the MAP estimate as the prior variance goes to zero. One could therefore pick a value of σ_0^2 as high as possible for mitigating overconfidence. However, this is undesirable since it also lowers the confidence of the training data and test data around them (i.e. the so-called *in-distribution data*), thus, causing underconfident predictions. Another common way to set this hyperparameter is by maximizing the validation log-likelihood (Ritter et al., 2018). This is also inadequate for our purpose since it only considers points close to the training data.

Inspired by Hendrycks et al. (2019) and Hein et al. (2019), we simultaneously prefer high confidence on the in-distribution validation set and low confidence (high entropy) on the out-of-distribution validation set. Let $\hat{\mathcal{D}} := \{\hat{\mathbf{x}}_i,\hat{t}_i\}_{i=1}^m$ be a validation set and $\tilde{\mathcal{D}} := \{\tilde{\mathbf{x}}_i\}_{i=1}^m$ be an out-of-distribution dataset. We then pick the optimal σ_0^2 by solving the following one-parameter optimization problem:

$$\arg\min_{\sigma_0^2} -\frac{1}{m}\sum_{i=1}^{m} \log p(y=\hat{t}_i|\hat{\mathbf{x}}_i,\mathcal{D}) + \lambda H[p(y|\tilde{\mathbf{x}}_i,\mathcal{D})]\,, \tag{8}$$

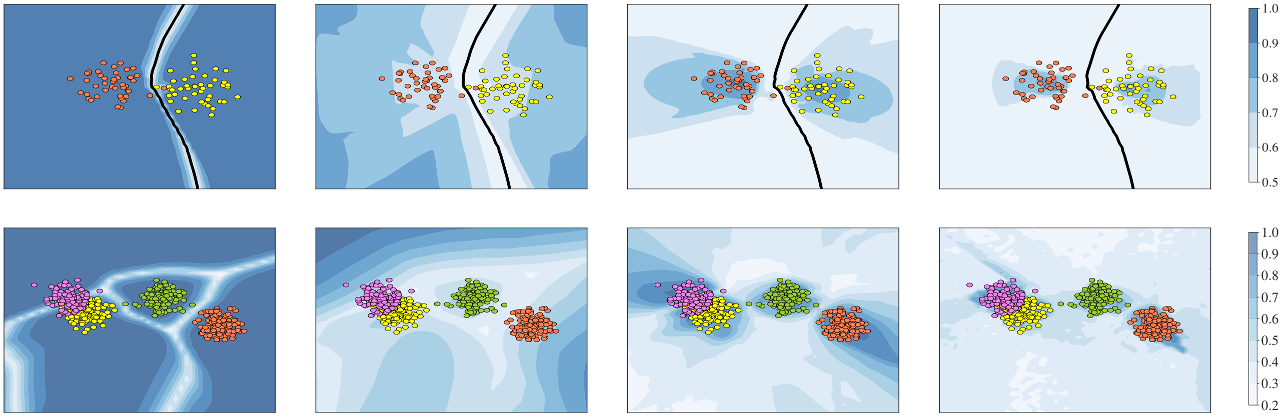
where $\lambda \in [0,1]$ is controlling the trade-off between both terms. The first term in (8) is the standard cross-entropy loss over $\hat{\mathcal{D}}$ while the second term is the negative predictive entropy over $\tilde{\mathcal{D}}$. Alternatively, the second term can be replaced by the cross-entropy loss where the target is the uniform probability vector. In all our experiments, we simply assume that $\tilde{\mathcal{D}}$ is a collection of uniform noise in the input space.

3. Related Work

The overconfidence problem of deep neural networks, and thus ReLU networks, has long been known in the deep learning community (Nguyen et al., 2015), although a formal description was only delivered recently. Many methods have been proposed to combat or at least detect this issue. *Post-hoc* heuristics based on temperature or Platt scaling (Platt et al., 1999; Guo et al., 2017; Liang et al., 2018) are unable to detect inputs with arbitrarily high confidence far away from the training data (Hein et al., 2019).

Many works on uncertainty quantification in deep learning have recently been proposed. Gast & Roth (2018) proposed lightweight probabilistic networks via assumed density filtering. Malinin & Gales (2018; 2019); Sensoy et al. (2018) employ a Dirichlet distribution to model the distribution of a network's output. Lakshminarayanan et al. (2017) quantify predictive uncertainty based on the idea of model ensembling and frequentist calibration. Hein et al. (2019) proposed enhanced training objectives based on robust optimization to mitigate this issue. Meinke & Hein (2020) proposed a similar approach with provable guarantees. However, they either lack in their theoretical analysis or do not employ probabilistic or Bayesian approximations. Our results, meanwhile, provide a theoretical justification to the commonly-used Gaussian approximations of NNs' weights, both Bayesian (Graves, 2011; Blundell et al., 2015; Louizos & Welling, 2016; Maddox et al., 2019, etc.) and non-Bayesian (Franchi et al., 2019; Lu et al., 2020, etc.).

Bayesian methods have long been thought to mitigate the



(a) MAP (b) Temp. scaling (c) LLLA (d) Full Laplace

Figure 4. Binary (top) and multi-class (bottom) toy classification problem. Background color represents confidence.

overconfidence problem on any neural network (MacKay, 1992a). Empirical evidence supporting this intuition has also been presented (Liu et al., 2019; Wu et al., 2019, etc.). Our results complement these with a theoretical justification for the ReLU-logistic case. Furthermore, our theoretical results show that, in some cases, an expensive Bayesian treatment over all layers of a network is not necessary (Theorem 2.4). Our results are thus theoretically validating the usage of Bayesian generalized linear models (Gelman et al., 2008) (especially in conjunction with ReLU features) and last-layer Bayesian methods (Snoek et al., 2015; Wilson et al., 2016a;b; Riquelme et al., 2018); and complementing the empirical analyses of Ober & Rasmussen (2019) and Brosse et al. (2020).

4. Experiments

We corroborate our theoretical results via four experiments using various Gaussian-based Bayesian methods. In Section 4.1 we visualize the confidence of 2D binary and multi-class toy datasets. In Section 4.2 we empirically validate our main result that the confidence of binary classification datasets approaches finite constants as δ increases. Furthermore, we show empirically that this property also holds in the multi-class case, along with the usefulness of Bayesian methods in standard OOD detection tasks in Section 4.3. Finally, in Section 4.4, we show that our results also hold in the case of last-layer Gaussian processes.

Unless stated otherwise, we use LeNet (for MNIST) or ResNet-18 (for CIFAR-10, SVHN, CIFAR-100) architectures. We train these networks by following the procedure described by Meinke & Hein (2020) (Appendix C). To obtain the optimal hyperparameter σ_0^2, we follow (8) with λ set to 0.25. We mainly use a *last-layer Laplace approximation* (LLLA)[3] where a Laplace approximation with an exact Hessian or its Kronecker factors is applied only to the last layer of a network (Appendix B). Whenever the approximations of predictive distribution in (4) and (5) cannot be used, we compute them via Monte Carlo integrations with 100 posterior samples. Other Laplace approximations that we use will be introduced in the subsection where they are first employed. Besides the vanilla MAP method, we use the temperature scaling method (Guo et al., 2017) as a baseline since it claims to give calibrated predictions in the frequentist sense. In particular, the optimal temperature is found via a validation log-likelihood maximization using PyCalib (Wenger et al., 2019). For each dataset that we use, we obtain a validation set via a random split from the respective test set.[4] Lastly, all numbers reported in this section are averages along with their standard deviations over 10 trials.

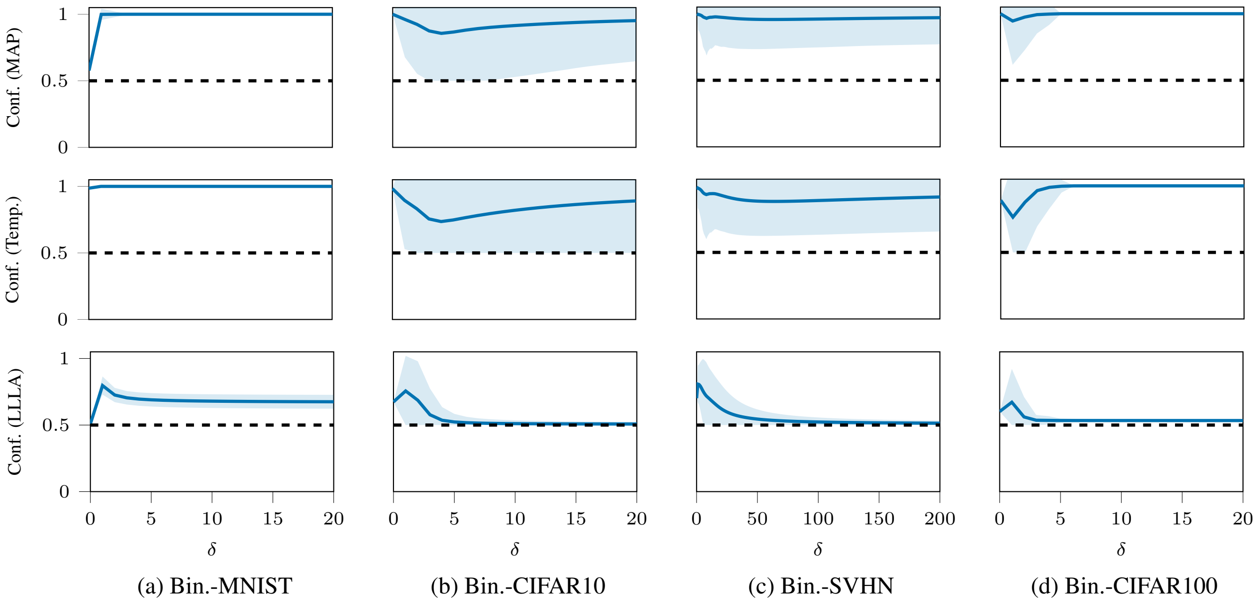
4.1. Toy Dataset

Here, the dataset is constructed by sampling the input points from k independent Gaussians. The corresponding targets indicate from which Gaussian the point was sampled. We use a 3-layer ReLU network with 20 hidden units at each layer. We use the exact Hessian and the full generalized-Gauss-Newton (GGN) approximation of the Hessian for the case of LLLA and all-layer Laplace approximations, respectively.

We show the results for the binary and multi-class cases in Figure 4. The MAP predictions have high confidence everywhere except at the region close to the decision boundary. Temperature scaling assigns low confidence to the training data, while assigning high confidence far away from

[3] `https://github.com/wiseodd/last_layer_laplace`.

[4] We use 50, 1000, and 2000 points for the toy, binary, and multi-class classification cases, respectively.



(a) Bin.-MNIST (b) Bin.-CIFAR10 (c) Bin.-SVHN (d) Bin.-CIFAR100

Figure 5. Confidence of MAP (top row), temperature scaling (middle row), and LLLA (bottom row) as functions of δ over the test sets of binary classification datasets. Thick blue lines and shades correspond to means and ± 3 standard deviations, respectively. Dashed lines signify the desirable confidence for δ sufficiently high.

them. LLLA, albeit simple, yields high confidence close to the training points and high uncertainty otherwise, while maintaining the MAP's decision boundary. Furthermore, we found that the all-layer Laplace approximation makes the aforementioned finding stronger: the boundaries of the high-confidence regions are now closer to the training data.

4.2. Binary Classification

We validate our theoretical finding by plotting the test confidence of various binary classification datasets as functions of δ. Each dataset is constructed by picking two classes which are most difficult to distinguish, based on the confusion matrix of the corresponding multi-class problem.

As shown in Figure 5, both MAP (top row) and temperature scaling (middle row) methods are overconfident for sufficiently large δ. Meanwhile, LLLA which represents Bayesian methods, mitigates this issue: As δ increases, the confidence converges to some constant close to the uniform confidence (one-half). Moreover, when $\delta = 1$ (the case of in-distribution data), LLLA retains higher confidence.

Table 1 further quantifies the results where we treat collections of 2000 uniform noise images scaled by $\delta = 100$ as the OOD datasets. Note that, while the resulting data points are not in the image space anymore, this construction is useful to assess the effectiveness of the Bayesian methods in unbounded problems. We report the standard metrics proposed by Hendrycks & Gimpel (2017): mean-maximum-confidence (MMC) and area-under-ROC-curve (AUR). Confirming our finding in Figure 5, LLLA is able to detect OOD data with high accuracy: for the chosen values of δ, the MMC and AUR values are close to the ideal values of 50 and 100, respectively. Both MAP and temperature scaling fail to do so since their confidence estimates saturate to one. These results (i) confirm our theoretical analysis in Section 2, (ii) show that even a simple Bayesian method yields good uncertainty estimates, and (iii) temperature scaling is not calibrated for outliers far-away from the training data.[5]

4.3. Multi-class Classification

We also show empirically that Bayesian methods yield a similar behavior in multi-class settings. On top of LLLA, representing Bayesian methods, we employ various other scalable Laplace approximation techniques: diagonal Laplace approximation (DLA) where a diagonal Gaussian is used to approximate the posterior over all layers of a network, and Kronecker-factored Laplace approximation (KFLA) (Ritter et al., 2018) where a matrix-variate normal is used to approximate the posterior over all layers. We use 20 posterior samples for both DLA and KFLA. We refer the reader to Appendix B for details.

For each training dataset we evaluate all methods both in the non-asymptotic (the corresponding OOD test datasets, e.g.

[5] This confirms the theoretical arguments of Hein et al. (2019).

Table 1. OOD detection for far-away points in binary classification settings. The in-distribution datasets are Binary-MNIST, Binary-CIFAR10, Binary-SVHN, and Binary-CIFAR100. Each OOD dataset is obtained by scaling uniform noise images in the corresponding input space of the in-distribution dataset with $\delta = 100$. All values are means and standard deviations over 10 trials.

	MAP		**+Temp.**		**+LLLA**	
	MMC ↓	AUR ↑	MMC ↓	AUR ↑	MMC ↓	AUR ↑
Binary-MNIST	99.9±0.0	-	100.0±0.0	-	79.4±0.9	-
Noise ($\delta = 100$)	100.0±0.0	0.2±0.1	100.0±0.0	45.1±5.8	**67.5±0.8**	**99.6±0.1**
Binary-CIFAR10	96.3±0.3	-	90.5±0.6	-	76.4±0.3	-
Noise ($\delta = 100$)	98.9±1.0	11.3±10.3	97.6±2.2	11.3±10.3	**50.6±0.1**	**99.5±0.1**
Binary-SVHN	99.4±0.0	-	98.2±0.1	-	80.7±0.1	-
Noise ($\delta = 100$)	98.8±0.6	50.5±42.3	95.9±3.0	50.5±42.3	**51.2±0.6**	**99.8±0.1**
Binary-CIFAR100	94.5±0.5	-	74.5±2.9	-	66.7±0.5	-
Noise ($\delta = 100$)	100.0±0.0	1.5±0.7	100.0±0.0	0.0±0.0	**53.5±0.1**	**93.6±1.8**

Table 2. Multi-class OOD detection results for MAP, last-layer Laplace (LLLA), (all-layers) diagonal Laplace (DLA), and (all-layers) Kronecker-Factored Laplace (KFLA). Each "far-away" Noise dataset is constructed as in Table 1 with $\delta = 2000$. All values are averages and standard deviations over 10 trials.

	MAP		**+Temp.**		**+LLLA**		**+DLA**		**+KFLA**	
	MMC ↓	AUR ↑	MMC ↓	AUR ↑	MMC ↓	AUR ↑	MMC ↓	AUR ↑	MMC ↓	AUR ↑
MNIST - MNIST	99.2±0.0	-	99.5±0.1	-	98.4±0.2	-	84.5±0.2	-	92.9±0.3	-
MNIST - EMNIST	82.3±0.0	89.2±0.1	87.6±1.4	88.9±0.2	70.2±1.9	**92.0±0.4**	**54.5±0.3**	87.7±0.4	58.7±0.4	89.6±0.3
MNIST - FMNIST	66.3±0.0	97.4±0.0	75.2±2.5	97.1±0.1	56.0±1.8	98.2±0.2	42.5±0.1	96.3±0.1	**39.9±0.5**	**98.6±0.1**
MNIST - Noise ($\delta = 2000$)	100.0±0.0	0.1±0.0	100.0±0.0	6.8±4.1	99.9±0.0	9.6±0.7	84.9±1.3	53.7±3.1	**55.6±2.0**	**97.3±0.4**
CIFAR10 - CIFAR10	97.1±0.1	-	95.4±0.2	-	92.8±1.1	-	88.4±0.1	-	86.5±0.1	-
CIFAR10 - SVHN	62.5±0.0	95.8±0.1	54.6±0.6	96.1±0.0	45.9±1.6	**96.4±0.1**	43.3±0.1	95.5±0.1	**43.0±0.1**	94.8±0.1
CIFAR10 - LSUN	74.5±0.0	91.9±0.1	66.9±0.6	92.2±0.1	57.4±1.9	92.7±0.4	49.0±0.5	**92.8±0.3**	**47.6±0.4**	92.2±0.2
CIFAR10 - Noise ($\delta = 2000$)	98.7±0.2	10.9±0.4	98.4±0.2	10.0±0.5	**17.4±0.0**	**100.0±0.0**	60.7±2.0	89.6±1.1	61.8±1.5	87.6±0.9
SVHN - SVHN	98.5±0.0	-	97.4±0.2	-	93.2±1.0	-	88.8±0.0	-	90.8±0.0	-
SVHN - CIFAR10	70.4±0.0	95.4±0.0	64.1±0.9	95.4±0.0	43.4±2.1	97.2±0.1	**38.0±0.1**	**97.6±0.0**	41.2±0.1	97.5±0.0
SVHN - LSUN	71.7±0.0	95.5±0.0	65.4±1.0	95.6±0.0	44.3±2.3	97.3±0.1	**39.5±0.7**	**97.5±0.2**	42.0±0.6	**97.5±0.1**
SVHN - Noise ($\delta = 2000$)	98.7±0.1	11.9±0.6	98.4±0.1	11.0±0.6	**27.5±0.1**	**99.6±0.0**	60.8±1.6	92.8±0.6	62.4±2.0	94.0±0.5
CIFAR100 - CIFAR100	81.2±0.1	-	78.9±0.8	-	74.6±0.2	-	76.4±0.2	-	73.4±0.2	-
CIFAR100 - SVHN	53.5±0.0	78.8±0.1	49.2±1.2	79.2±0.1	42.7±0.3	**80.4±0.2**	46.0±0.1	79.6±0.2	**41.4±0.1**	80.1±0.2
CIFAR100 - LSUN	50.7±0.0	81.0±0.1	46.8±1.1	81.1±0.1	39.8±0.2	**82.6±0.2**	43.5±0.3	81.5±0.2	**39.7±0.4**	81.6±0.3
CIFAR100 - Noise ($\delta = 2000$)	99.5±0.1	2.8±0.2	99.4±0.1	2.6±0.2	**5.9±0.0**	**99.9±0.0**	41.5±1.5	84.2±0.9	37.1±1.3	84.2±0.8

SVHN and LSUN for CIFAR-10) and asymptotic (Noise datasets) regime. Each "far-away" Noise dataset is constructed by scaling 2000 uniform noise images in the corresponding input space with $\delta = 2000$. As in the previous section, we report the MMC and AUR metrics.

As presented in Table 2, all the Bayesian methods improve the OOD detection performance of the base models both in the non-asymptotic and asymptotic regime. Especially in the asymptotic regime, all the Bayesian methods perform well, empirically confirming our hypothesis that our theoretical analysis carries over to the multi-class setting. Meanwhile, both MAP's and temperature scaling's MMC and AUR are close to 100 and 0, respectively.[6] Moreover, while LLLA is the simplest Bayesian method in this experiment, it often outperforms DLA and KFLA. Our finding agrees with the prior observation that last-layer Bayesian approximations are often sufficient (Ober & Rasmussen, 2019; Brosse et al., 2020). Furthermore, in Appendix D we also show that we can apply Laplace approximations on top of prior OOD detection methods such as ACET (Hein et al., 2019) and Outlier-Exposure (Hendrycks et al., 2019) to achieve state-of-the-art performance in the non-asymptotic regime, while also having high uncertainty in the asymptotic regime.

In Table 3, we present the computational cost analysis in terms of wall-clock time. We measure the time required for each method to do posterior inference (or finding the optimal temperature) and to make predictions. While MAP and temperature scaling are fast, as we have shown in the previous results, they are overconfident. Among the Bayesian methods, since the cost of LLLA is constant w.r.t. the network depth, we found that it is up to two orders of magnitude faster than DLA and KFLA when making predictions. All in all, this finding, combined with the previous results, makes this simple Bayesian method attractive in applications.

[6] I.e. the worst values for those metrics.

Table 3. Wall-clock time (in second) of posterior inferences and predictions over test sets.

	MNIST	**CIFAR-10**	**SVHN**	**CIFAR-100**
Inference				
MAP	-	-	-	-
+Temp.	0.0	0.0	0.0	0.0
+LLLA	1.8	23.0	33.7	23.1
+DLA	1.3	22.9	33.6	23.0
+KFLA	4.3	78.1	115.1	78.6
Prediction				
MAP	0.4	1.2	2.7	1.2
+Temp.	0.4	1.2	2.7	1.2
+LLLA	0.9	1.7	4.3	1.6
+DLA	7.3	100.0	260.6	100.4
+KFLA	21.5	151.0	392.8	151.7

Table 4. Multi-class OOD detection results for deep kernel learning (DKL). Each "far-away" Noise dataset is constructed as in Table 2 with $\delta = 2000$. All values are averages and standard deviations over 10 trials.

Train - Test	**MMC** ↓	**AUR** ↑
MNIST - MNIST	99.6±0.0	-
MNIST - EMNIST	83.9±0.0	94.4±0.1
MNIST - FMNIST	70.6±0.1	98.8±0.0
MNIST - Noise	58.6±0.5	99.7±0.0
CIFAR10 - CIFAR10	97.5±0.0	-
CIFAR10 - SVHN	50.6±0.1	98.6±0.0
CIFAR10 - LSUN	77.9±0.3	93.4±0.1
CIFAR10 - Noise	56.5±0.7	98.5±0.1
SVHN - SVHN	98.6±0.0	-
SVHN - CIFAR10	72.7±0.0	96.0±0.0
SVHN - LSUN	76.7±0.1	95.1±0.1
SVHN - Noise	48.6±0.7	99.4±0.0
CIFAR100 - CIFAR100	80.5±0.0	-
CIFAR100 - SVHN	72.7±0.1	63.1±0.1
CIFAR100 - LSUN	66.8±0.4	69.7±0.3
CIFAR100 - Noise	43.1±1.1	90.3±0.7

4.4. Last-layer Gaussian Processes

It has been shown that Gaussian-approximated linear models with infinitely many features, e.g. two-layer networks with infinitely wide hidden layers, are equivalent to Gaussian processes (Neal, 1996). In the language of Theorem 2.4, this is the case when the dimension of the feature space $\mathbb{R}^d$ goes to infinity. It is therefore interesting to see, at least empirically, whether low asymptotic confidence is also attained in this case.

While unlike LLLA, deep kernel learning (DKL, Wilson et al., 2016a;b) is not a *post-hoc* method, it is a suitable model for showcasing our theory in the case of last-layer Gaussian processes. We therefore train stochastic variational DKL models (Wilson et al., 2016b) which use the same networks used in the previous experiment (minus the top layer) as their feature extractors, following the implementation provided by GPyTorch (Gardner et al., 2018). The training protocol is identical as before (cf. Appendix C). To compute each prediction, we use 20 samples from the Gaussian process posterior. We are mainly interested in the performance of DKL in term of multi-class OOD detection (both in asymptotic and non-asymptotic regimes), similar to the previous section.

The results are presented in Table 4. When compared to the results of the MAP estimation in Table 2, we found that DKL is able to mitigate asymptotic overconfidence (see results against the Noise dataset). These results empirically verify that our analysis also holds in the non-parametric infinite-width regime. Nevertheless, we found that LLLA generally outperforms DKL both in term of MMC and AUR metrics. This finding, along with the simplicity and efficiency of LLLA make it more attractive than DKL, especially since DKL requires retraining and thus cannot simply be applied to pre-trained ReLU networks.

5. Conclusion

We have shown analytically that Gaussian approximations of weights distributions, when applied on binary ReLU classification networks, can mitigate the asymptotic overconfidence problem that plagues deep learning. While this behavior does not seem surprising—indeed, Gaussian-based approximate Bayesian methods have empirically been known to give good uncertainty estimates—formal statements regarding this property had been missing. Our results provide some of these statements. Furthermore, we have shown, both theoretically and empirically, that a sufficient condition for good uncertainty estimates in ReLU networks is to be "a bit Bayesian": apply a Gaussian-based probabilistic method, in particular a Bayesian one, to the last layer of the network. Our analysis further validates the common usage of approximate inference methods—both Bayesian and non-Bayesian—which leverage the Gaussian distribution.

Acknowledgements

The authors gratefully acknowledge financial support by the European Research Council through ERC StG Action 757275 / PANAMA; the DFG Cluster of Excellence "Machine Learning - New Perspectives for Science", EXC 2064/1, project number 390727645; the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A); and funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg. AK is grateful to Alexander Meinke for the pre-trained models and the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for support. AK also thanks all members of Methods of Machine Learning group for helpful feedback.

References

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. Understanding deep neural networks with rectified linear units. In *ICLR*, 2018.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. In *ICML*, 2015.

Brier, G. W. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1), 1950.

Brosse, N., Riquelme, C., Martin, A., Gelly, S., and Moulines, É. On last-layer algorithms for classification: Decoupling representation from uncertainty estimation. *arXiv preprint arXiv:2001.08049*, 2020.

Dangel, F., Kunstner, F., and Hennig, P. BackPACK: Packing more into Backprop. In *ICLR*, 2020.

Franchi, G., Bursuc, A., Aldea, E., Dubuisson, S., and Bloch, I. TRADI: Tracking deep neural network weight distributions. *arXiv preprint arXiv:1912.11316*, 2019.

Gal, Y. Uncertainty in deep learning. *University of Cambridge*, 2016.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration. In *NIPS*, 2018.

Gast, J. and Roth, S. Lightweight probabilistic deep networks. In *CVPR*, 2018.

Gelman, A., Jakulin, A., Pittau, M. G., Su, Y.-S., et al. A weakly informative default prior distribution for logistic and other regression models. *The annals of applied statistics*, 2(4), 2008.

Gibbs, M. *Bayesian Gaussian processes for regression and classification*. PhD thesis, University of Cambridge, 1998.

Graves, A. Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems 24*, pp. 2348–2356. 2011.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *ICML*, 2017.

Gupta, A. K. and Nagar, D. K. *Matrix variate distributions*. Chapman and Hall/CRC, 1999.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hein, M., Andriushchenko, M., and Bitterwolf, J. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *CVPR*, 2019.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017.

Hendrycks, D., Mazeika, M., and Dietterich, T. Deep anomaly detection with outlier exposure. In *ICLR*, 2019.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *CVPR*, 2017.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, 2017.

Liang, S., Li, Y., and Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR*, 2018.

Liu, X., Li, Y., Wu, C., and Hsieh, C.-J. Adv-BNN: Improved adversarial defense through robust bayesian neural network. In *ICLR*, 2019.

Louizos, C. and Welling, M. Structured and efficient variational deep learning with matrix gaussian posteriors. In *ICML*, 2016.

Louizos, C. and Welling, M. Multiplicative normalizing flows for variational Bayesian neural networks. In *ICML*, 2017.

Lu, Z., Ie, E., and Sha, F. Uncertainty Estimation with Infinitesimal Jackknife, Its Distribution and Mean-Field Approximation. *arXiv preprint arXiv:2006.07584*, 2020.

MacKay, D. J. The evidence framework applied to classification networks. *Neural computation*, 1992a.

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

MacKay, D. J. Probable networks and plausible predictions—a review of practical bayesian methods for supervised neural networks. *Network: computation in neural systems*, 1995.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. A simple baseline for bayesian uncertainty in deep learning. In *NeurIPS*, 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Malinin, A. and Gales, M. Predictive uncertainty estimation via prior networks. In *NIPS*, 2018.

Malinin, A. and Gales, M. Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness. In *NIPS*, 2019.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417, 2015.

Meinke, A. and Hein, M. Towards neural networks that provably know when they don't know. In *ICLR*, 2020.

Naeini, M. P., Cooper, G., and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, 2015.

Neal, R. M. Priors for infinite networks. In *Bayesian Learning for Neural Networks*. Springer, 1996.

Nguyen, A., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, 2015.

Ober, S. W. and Rasmussen, C. E. Benchmarking the neural linear model for regression. *arXiv preprint arXiv:1912.08416*, 2019.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *NeurIPS*, 2019.

Platt, J. et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Riquelme, C., Tucker, G., and Snoek, J. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. In *ICLR*, 2018.

Ritter, H., Botev, A., and Barber, D. A scalable laplace approximation for neural networks. In *ICLR*, 2018.

Sensoy, M., Kaplan, L., and Kandemir, M. Evidential deep learning to quantify classification uncertainty. In *NIPS*, 2018.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M., and Adams, R. Scalable bayesian optimization using deep neural networks. In *ICML*, 2015.

Spiegelhalter, D. J. and Lauritzen, S. L. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 1990.

Wenger, J., Kjellström, H., and Triebel, R. Nonparametric calibration for classification. *arXiv preprint arXiv:1906.04933*, 2019.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *AISTATS*, 2016a.

Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P. Stochastic variational deep kernel learning. In *NIPS*, 2016b.

Wu, A., Nowozin, S., Meeds, E., Turner, R. E., Hernandez-Lobato, J. M., and Gaunt, A. L. Deterministic variational inference for robust bayesian neural networks. In *ICLR*, 2019.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. Noisy natural gradient as variational inference. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5852–5861, 2018.

A. Proofs

In the followings, the norm $\|\cdot\|$ is the standard ℓ^2-norm.

Proposition 2.2 (Invariance property)**.** *Let $f_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}$ be a binary classifier network parametrized by $\boldsymbol{\theta}$ and let $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be the distribution over $\boldsymbol{\theta}$. Then for any $\mathbf{x} \in \mathbb{R}^n$, we have $\sigma(f_{\boldsymbol{\mu}}(\mathbf{x})) = 0.5$ if and only if $\sigma(z(\mathbf{x})) = 0.5$.*

Proof. Let $\mathbf{x} \in \mathbb{R}^n$ be arbitrary and denote $\mu_f := f_{\boldsymbol{\theta}_{\text{MAP}}}(\mathbf{x})$ and $v_f := \mathbf{d}(\mathbf{x})^\top \boldsymbol{\Sigma} \mathbf{d}(\mathbf{x})$. For the forward direction, suppose that $\sigma(\mu_f) = 0.5$. This implies that $\mu_f = 0$, and we have $\sigma(0/(1 + \pi/8\, v_f)^{1/2}) = \sigma(0) = 0.5$. For the reverse direction, suppose that $\sigma(\mu_f/(1 + \pi/8\, v_f)^{1/2}) = 0.5$. This implies $\mu_f/(1 + \pi/8\, v_f)^{1/2} = 0$. Since the denominator of the l.h.s. is positive, it follows that μ_f must be 0, implying that $\sigma(\mu_f) = 0.5$. □

Lemma A.1 (Hein et al., 2019)**.** *Let $\{Q_i\}_{l=1}^R$ be the set of linear regions associated to the ReLU network $f : \mathbb{R}^n \to \mathbb{R}^k$. For any $\mathbf{x} \in \mathbb{R}^n$ there exists an $\alpha > 0$ and $t \in \{1, \dots, R\}$ such that $\delta\mathbf{x} \in Q_t$ for all $\delta \geq \alpha$. Furthermore, the restriction of f to Q_t can be written as an affine function $\mathbf{U}^\top\mathbf{x} + \mathbf{c}$ for some suitable $\mathbf{U} \in \mathbb{R}^{n\times k}$ and $\mathbf{c} \in \mathbb{R}^k$.* □

Theorem 2.3 (All-layer approximation)**.** *Let $f_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}$ be a binary ReLU classification network parametrized by $\boldsymbol{\theta} \in \mathbb{R}^p$ with $p \geq n$, and let $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be the Gaussian approximation over the parameters. Then for any input $\mathbf{x} \in \mathbb{R}^n$,*

$$\lim_{\delta\to\infty} \sigma(|z(\delta\mathbf{x})|) \leq \sigma\left(\frac{\|\mathbf{u}\|}{s_{\text{min}}(\mathbf{J})\sqrt{\pi/8\,\lambda_{\text{min}}(\boldsymbol{\Sigma})}} \right), \quad (6)$$

where $\mathbf{u} \in \mathbb{R}^n$ is a vector depending only on $\boldsymbol{\mu}$ and the $n \times p$ matrix $\mathbf{J} := \frac{\partial \mathbf{u}}{\partial \boldsymbol{\theta}}\big|_{\boldsymbol{\mu}}$ is the Jacobian of $\mathbf{u}$ w.r.t. $\boldsymbol{\theta}$ at $\boldsymbol{\mu}$. Moreover, if $f_{\boldsymbol{\theta}}$ has no bias parameters, then there exists $\alpha > 0$ such that for any $\delta \geq \alpha$, we have that

$$\sigma(|z(\delta\mathbf{x})|) \leq \lim_{\delta\to\infty} \sigma(|z(\delta\mathbf{x})|)\,.$$

Proof. By Lemma 3.1 of Hein et al. (2019) (also presented in Lemma A.1) there exists an $\alpha > 0$ and a linear region R, along with $\mathbf{u} \in \mathbb{R}^n$ and $c \in \mathbb{R}$, such that for any $\delta \geq \alpha$, we have that $\delta\mathbf{x} \in R$ and the restriction $f_{\boldsymbol{\theta}}|_R$ can be written as $\mathbf{u}^\top\mathbf{x} + c$. Note that, for any such δ, the vector $\mathbf{u}$ and scalar c are constant w.r.t. $\delta\mathbf{x}$. Therefore for any such δ, we can write the gradient $\mathbf{d}(\delta\mathbf{x})$ as follows:

$$\begin{aligned} \mathbf{d}(\delta\mathbf{x}) &= \frac{\partial(\delta\mathbf{u}^\top\mathbf{x})}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\mu}} + \frac{\partial c}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\mu}} = \delta \frac{\partial\mathbf{u}}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\mu}}^\top \mathbf{x} + \frac{\partial c}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\mu}} \\ &= \delta\left(\mathbf{J}^\top\mathbf{x} + \frac{1}{\delta}\nabla_{\boldsymbol{\theta}}\, c|_{\boldsymbol{\mu}}\right). \end{aligned} \quad (9)$$

Hence, by (5),

$$\begin{aligned} |z(\delta\mathbf{x})| &= \frac{|\delta\mathbf{u}^\top\mathbf{x} + c|}{\sqrt{1 + \pi/8\,\mathbf{d}(\delta\mathbf{x})^\top\boldsymbol{\Sigma}\mathbf{d}(\delta\mathbf{x})}} \\ &= \frac{|\delta(\mathbf{u}^\top\mathbf{x} + \frac{1}{\delta}c)|}{\sqrt{1 + \pi/8\,\delta^2(\mathbf{J}^\top\mathbf{x} + \frac{1}{\delta}\nabla_{\boldsymbol{\theta}}\, c|_{\boldsymbol{\mu}})^\top\boldsymbol{\Sigma}(\mathbf{J}^\top\mathbf{x} + \frac{1}{\delta}\nabla_{\boldsymbol{\theta}}\, c|_{\boldsymbol{\mu}})}} \\ &= \frac{\cancel{\delta}|(\mathbf{u}^\top\mathbf{x} + \frac{1}{\delta}c)|}{\cancel{\delta}\sqrt{\frac{1}{\delta^2} + \pi/8\,(\mathbf{J}^\top\mathbf{x} + \frac{1}{\delta}\nabla_{\boldsymbol{\theta}}\, c|_{\boldsymbol{\mu}})^\top\boldsymbol{\Sigma}(\mathbf{J}^\top\mathbf{x} + \frac{1}{\delta}\nabla_{\boldsymbol{\theta}}\, c|_{\boldsymbol{\mu}})}} \end{aligned}$$

Now, notice that as $\delta \to \infty$, $1/\delta^2$ and $1/\delta$ goes to zero. So, in the limit, we have that

$$\lim_{\delta\to\infty} |z(\delta\mathbf{x})| = \frac{|\mathbf{u}^\top\mathbf{x}|}{\sqrt{\pi/8\,(\mathbf{J}^\top\mathbf{x})^\top\boldsymbol{\Sigma}(\mathbf{J}^\top\mathbf{x})}}\,.$$

Using the Cauchy-Schwarz inequality and Lemma A.3, we can upper-bound this limit with

$$\lim_{\delta\to\infty} |z(\delta\mathbf{x})| \leq \frac{\|\mathbf{u}\|\|\mathbf{x}\|}{\sqrt{\pi/8\,\lambda_{\text{min}}(\boldsymbol{\Sigma})\|\mathbf{J}^\top\mathbf{x}\|^2}}\,.$$

The following lemma is needed to get the desired result.

Lemma A.2. *Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{z} \in \mathbb{R}^n$ with $m \geq n$, then $\|\mathbf{A}\mathbf{z}\|^2 \geq s^2_{\text{min}}(\mathbf{A})\|\mathbf{z}\|^2$.*

Proof. By SVD, $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$. Notice that $\mathbf{U}, \mathbf{V}$ are orthogonal and thus are isometries, and that $\mathbf{S}$ is a rectangular diagonal matrix with n non-zero elements. Therefore,

$$\begin{aligned} \|\mathbf{U}(\mathbf{S}\mathbf{V}^\top\mathbf{z})\|^2 = \|\mathbf{S}\mathbf{V}^\top\mathbf{z}\|^2 &= \sum_{i=1}^n s_i^2(\mathbf{A})(\mathbf{V}^\top\mathbf{z})_i^2 \\ &\geq s^2_{\text{min}}(\mathbf{A})\sum_{i=1}^n (\mathbf{V}^\top\mathbf{z})_i^2 \\ &= s^2_{\text{min}}(\mathbf{A})\|\mathbf{V}^\top\mathbf{z}\|^2 = s^2_{\text{min}}(\mathbf{A})\|\mathbf{z}\|^2\,, \end{aligned} \quad (10)$$

thus the proof is complete. □

Notice that $\mathbf{J}^\top \in \mathbb{R}^{p\times n}$ with $p \geq n$ by our hypothesis. Therefore, using the previous lemma on $\|\mathbf{J}^\top\mathbf{x}\|^2$ in conjunction with $s_{\text{min}}(\mathbf{J}) = s_{\text{min}}(\mathbf{J}^\top)$, we conclude that

$$\begin{aligned} \lim_{\delta\to\infty} |z(\delta\mathbf{x})| &\leq \frac{\|\mathbf{u}\|\|\mathbf{x}\|}{\sqrt{\pi/8\,\lambda_{\text{min}}(\boldsymbol{\Sigma})\, s^2_{\text{min}}(\mathbf{J})\|\mathbf{x}\|^2}} \\ &= \frac{\|\mathbf{u}\|}{s_{\text{min}}(\mathbf{J})\sqrt{\pi/8\,\lambda_{\text{min}}(\boldsymbol{\Sigma})}}\,, \end{aligned} \quad (11)$$

thus the first result is proved.

To prove the second statement, let $L := \lim_{\delta\to\infty} |z(\delta\mathbf{x})|$. Since L is the limit of $|z|_Q(\delta\mathbf{x})|$ in the linear region Q given by Lemma A.1, it is sufficient to show that the function $(0, \infty] \to \mathbb{R}$ defined by $\delta \mapsto |z|_Q(\delta \mathbf{x})|$ is increasing.

For some suitable choices of $\mathbf{u} \in \mathbb{R}^n$ that depends on $\boldsymbol{\mu}$, we can write the restriction of the "point-estimated" ReLU network $f_{\boldsymbol{\mu}}|_Q(\mathbf{x})$ as $\mathbf{u}^\top \mathbf{x}$ by definition of ReLU network and since we assume that f has no bias parameters. Furthermore, we let the matrix $\mathbf{J} := \frac{\partial \mathbf{u}}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\mu}}$ to be the Jacobian of $\mathbf{u}$ w.r.t. $\boldsymbol{\theta}$ at $\boldsymbol{\mu}$. Therefore for any $\delta \geq \alpha$, we can write as a function of δ:

$$\begin{aligned} |z|_Q(\delta \mathbf{x})| &= \frac{|\delta \mathbf{u}^\top \mathbf{x}|}{\sqrt{1 + \pi/8 \, \delta^2 \, (\mathbf{J}^\top \mathbf{x})^\top \boldsymbol{\Sigma} (\mathbf{J}^\top \mathbf{x})}} \\ &=: \frac{|\delta a|}{\sqrt{1 + \pi/8 \, \delta^2 \, b}} \,, \end{aligned}$$

where for simplicity we have let $a := \mathbf{u}^\top \mathbf{x}$ and $b := (\mathbf{J}^\top \mathbf{x})^\top \boldsymbol{\Sigma} (\mathbf{J}^\top \mathbf{x})$. The derivative is therefore given by

$$\frac{d}{d\delta} |z|_Q(\delta \mathbf{x})| = \frac{\delta |a|}{(1 + \delta^2 b)^{\frac{3}{2}} |\delta|}$$

and since $\boldsymbol{\Sigma}$ is positive-definite, it is non-negative for $\delta \in (0, \infty]$. Thus we conclude that $|z|_Q(\delta \mathbf{x})|$ is an increasing function. □

Theorem 2.4 (Last-layer approximation)**.** *Let $g : \mathbb{R}^d \to \mathbb{R}$ be a binary linear classifier defined by $g(\phi(\mathbf{x})) := \mathbf{w}^\top \phi(\mathbf{x})$ where $\phi : \mathbb{R}^n \to \mathbb{R}^d$ is a fixed ReLU network and let $\mathcal{N}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be the Gaussian approximation over the last-layer's weights. Then for any input $\mathbf{x} \in \mathbb{R}^n$,*

$$\lim_{\delta \to \infty} \sigma(|z(\delta \mathbf{x})|) \leq \sigma \left(\frac{\|\boldsymbol{\mu}\|}{\sqrt{\pi/8 \, \lambda_{\text{min}}(\boldsymbol{\Sigma})}} \right) . \qquad (7)$$

Moreover, if ϕ has no bias parameters, then there exists $\alpha > 0$ such that for any $\delta \geq \alpha$, we have that

$$\sigma(|z(\delta \mathbf{x})|) \leq \lim_{\delta \to \infty} \sigma(|z(\delta \mathbf{x})|) \, .$$

Proof. By Lemma 3.1 of Hein et al. (2019) there exists $\alpha > 0$ and a linear region R, along with $\mathbf{U} \in \mathbb{R}^{d \times n}$ and $\mathbf{c} \in \mathbb{R}^d$, such that for any $\delta \geq \alpha$, we have that $\delta \mathbf{x} \in R$ and the restriction $\phi|_R$ can be written as $\mathbf{U}\mathbf{x} + \mathbf{c}$. Therefore, for any such δ,

$$\begin{aligned} |z \circ \phi|_R(\delta \mathbf{x})| &= \frac{|\boldsymbol{\mu}^\top (\delta \mathbf{U}\mathbf{x} + \mathbf{c})|}{\sqrt{1 + \pi/8 \, (\delta \mathbf{U}\mathbf{x} + \mathbf{b})^\top \boldsymbol{\Sigma} (\delta \mathbf{U}\mathbf{x} + \mathbf{c})}} \\ &= \frac{|\boldsymbol{\mu}^\top (\mathbf{U}\mathbf{x} + \frac{1}{\delta}\mathbf{c})|}{\sqrt{\frac{1}{\delta^2} + \pi/8 \, (\mathbf{U}\mathbf{x} + \frac{1}{\delta}\mathbf{c})^\top \boldsymbol{\Sigma} (\mathbf{U}\mathbf{x} + \frac{1}{\delta}\mathbf{c})}} \end{aligned}$$

Now, notice that as $\delta \to \infty$, $1/\delta^2$ and $1/\delta$ goes to zero. So, in the limit, we have that

$$\lim_{\delta \to \infty} |z \circ \phi|_R(\delta \mathbf{x})| = \frac{|\boldsymbol{\mu}^\top (\mathbf{U}\mathbf{x})|}{\sqrt{\pi/8 \, (\mathbf{U}\mathbf{x})^\top \boldsymbol{\Sigma} (\mathbf{U}\mathbf{x})}} \, .$$

We need the following lemma to obtain the bound.

Lemma A.3. *Let $\mathbf{x} \in \mathbb{R}^n$ be a vector and $\mathbf{A} \in \mathbb{R}^{n \times n}$ be an SPD matrix. If $\lambda_{\text{min}}(\mathbf{A})$ is the minimum eigenvalue of $\mathbf{A}$, then $\mathbf{x}^\top \mathbf{A} \mathbf{x} \geq \lambda_{\text{min}} \|\mathbf{x}\|^2$.*

Proof. Since $\mathbf{A}$ is SPD, it admits an eigendecomposition $\mathbf{A} = \mathbf{Q} \boldsymbol{\Lambda} \mathbf{Q}^\top$ and $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{\Lambda}^{\frac{1}{2}}$ makes sense. Therefore, by keeping in mind that $\mathbf{Q}^\top \mathbf{x}$ is a vector in $\mathbb{R}^n$, we have

$$\begin{aligned} \mathbf{x}^\top \mathbf{A} \mathbf{x} = \mathbf{x}^\top \mathbf{Q} \boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{\Lambda}^{\frac{1}{2}} \mathbf{Q}^\top \mathbf{x} &= \|\boldsymbol{\Lambda}^{\frac{1}{2}} \mathbf{Q}^\top \mathbf{x}\|^2 \\ &= \sum_{i=1}^n \lambda_i(\mathbf{A}) (\mathbf{Q}^\top \mathbf{x})_i^2 \geq \lambda_{\text{min}}(\mathbf{A}) \sum_{i=1}^n (\mathbf{Q}^\top \mathbf{x})_i^2 \\ &= \lambda_{\text{min}}(\mathbf{A}) \|\mathbf{Q}^\top \mathbf{x}\|^2 = \lambda_{\text{min}}(\mathbf{A}) \|\mathbf{x}\|^2 \,, \end{aligned}$$

where the last equality is obtained since $\|\mathbf{Q}^\top \mathbf{x}\|^2 = \mathbf{x}^\top \mathbf{Q}^\top \mathbf{Q} \mathbf{x}$ and by noting that $\mathbf{Q}$ is an orthogonal matrix. □

Using the Cauchy-Schwarz inequality and the previous lemma, we can upper-bound the limit with

$$\begin{aligned} \lim_{\delta \to \infty} |z \circ \phi|_R(\delta \mathbf{x})| &\leq \frac{\|\boldsymbol{\mu}\| \|\mathbf{U}\mathbf{x}\|}{\sqrt{\pi/8 \, \lambda_{\text{min}}(\boldsymbol{\Sigma}) \|\mathbf{U}\mathbf{x}\|^2}} \\ &= \frac{\|\boldsymbol{\mu}\|}{\sqrt{\pi/8 \, \lambda_{\text{min}}(\boldsymbol{\Sigma})}} \,, \end{aligned}$$

which concludes the proof for the first statement.

For the second statement, since the previous limit is the limit of $|z|_R(\delta \mathbf{x})|$ in the linear region R, it is sufficient to show that the function $(0, \infty] \to \mathbb{R}$ defined by $\delta \mapsto |z|_R(\delta \mathbf{x})|$ is increasing. For some $\mathbf{U} \in \mathbb{R}^{d \times n}$ that depends on the fixed parameter of ϕ, we write the restriction $\phi|_R(\mathbf{x})$ as $\mathbf{U}\mathbf{x}$ by definition of ReLU network and since ϕ is assumed to have no bias parameters. Therefore for any $\delta \geq \alpha$, we can write as a function of δ:

$$\begin{aligned} |z|_Q(\delta \mathbf{x})| &= \frac{|\delta \boldsymbol{\mu}^\top \mathbf{U}\mathbf{x}|}{\sqrt{1 + \pi/8 \, \delta^2 \, (\mathbf{U}\mathbf{x})^\top \boldsymbol{\Sigma} (\mathbf{U}\mathbf{x})}} \\ &=: \frac{|\delta a|}{\sqrt{1 + \pi/8 \, \delta^2 \, b}} \,, \end{aligned}$$

where for simplicity we have let $a := \boldsymbol{\mu}^\top \mathbf{U}\mathbf{x}$ and $b := (\mathbf{U}\mathbf{x})^\top \boldsymbol{\Sigma} (\mathbf{U}\mathbf{x})$. The derivative is therefore given by

$$\frac{d}{d\delta} |z|_Q(\delta \mathbf{x})| = \frac{\delta |a|}{(1 + \delta^2 b)^{\frac{3}{2}} |\delta|}$$

and since $\boldsymbol{\Sigma}$ is positive-definite, it is non-negative for $\delta \in (0, \infty]$. Thus we conclude that $|z|_Q(\delta \mathbf{x})|$ is an increasing function. □

Proposition 2.5 (All-layer Laplace)**.** *Let $f_{\boldsymbol{\theta}}$ be a binary ReLU classification network modeling a Bernoulli distribution $p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{B}(\sigma(f_{\boldsymbol{\theta}}(\mathbf{x})))$ with parameter $\boldsymbol{\theta} \in \mathbb{R}^p$. Let $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be the posterior obtained via a Laplace approximation with prior $\mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \sigma_0^2\mathbf{I})$, $\mathbf{H}$ be the Hessian of the negative log-likelihood at $\boldsymbol{\mu}$, and $\mathbf{J}$ be the Jacobian as in Theorem 2.3. Then for any input $\mathbf{x} \in \mathbb{R}^n$, the confidence $\sigma(|z(\mathbf{x})|)$ is a decreasing function of σ_0^2 with limits*

$$\lim_{\sigma_0^2 \to \infty} \sigma(|z(\mathbf{x})|) \leq \sigma\left(\frac{|f_{\boldsymbol{\mu}}(\mathbf{x})|}{1 + \sqrt{\pi/8}\, \lambda_{\text{max}}(\mathbf{H}) \|\mathbf{J}\mathbf{x}\|^2}\right)$$
$$\lim_{\sigma_0^2 \to 0} \sigma(|z(\mathbf{x})|) = \sigma(|f_{\boldsymbol{\mu}}(\mathbf{x})|) \, .$$

Proof. The assumption on the prior implies that $-\log p(\boldsymbol{\theta}) = 1/2\, \boldsymbol{\theta}^\top (1/\sigma_0^2 \mathbf{I}) \boldsymbol{\theta} + \text{const}$, which has Hessian $1/\sigma_0^2 \mathbf{I}$. Thus, the Hessian of the negative log posterior $-\log p(\boldsymbol{\theta}|\mathcal{D}) = -\log p(\boldsymbol{\theta}) - \log \prod_{\mathbf{x},t \in \mathcal{D}} p(y|\mathbf{x}, \boldsymbol{\theta})$ is $1/\sigma_0^2 \mathbf{I} + \mathbf{H}$. This implies that the posterior covariance $\boldsymbol{\Sigma}$ of the Laplace approximation is given by

$$\boldsymbol{\Sigma} = \left(\frac{1}{\sigma_0^2}\mathbf{I} + \mathbf{H}\right)^{-1} . \tag{12}$$

Therefore, the ith eigenvalue of $\boldsymbol{\Sigma}$ for any $i = 1, \dots, n$ is

$$\lambda_i(\boldsymbol{\Sigma}) = \frac{1}{1/\sigma_0^2 + \lambda_i(\mathbf{H})} = \frac{\sigma_0^2}{1 + \sigma_0^2 \lambda_i(\mathbf{H})} \, .$$

For all $i = 1, \dots, n$, the derivative of $\lambda_i(\boldsymbol{\Sigma})$ w.r.t. σ_0^2 is $1/(1 + \sigma_0^2 \lambda_i(\mathbf{H}))^2$ which is non-negative. This tells us that $\lambda_i(\boldsymbol{\Sigma})$ is a non-decreasing function of σ_0^2. Furthermore, it is also clear that $\sigma_0^2/(1 + \sigma_0^2\lambda_i(\mathbf{H}))$ goes to $1/\lambda_i(\mathbf{H})$ as σ_0^2 goes to infinity, while it goes to 0 as σ_0^2 goes to zero.

Now, we can write.

$$|z(\mathbf{x})| = \frac{|f_{\boldsymbol{\mu}}(\mathbf{x})|}{\sqrt{1 + \pi/8 \sum_{i=1}^d \lambda_i(\boldsymbol{\Sigma}) (\mathbf{Q}^\top \mathbf{d})_i^2}} \, , \tag{13}$$

where $\boldsymbol{\Sigma} = \mathbf{Q} \operatorname{diag}(\lambda_i(\boldsymbol{\Sigma}), \dots, \lambda_d(\boldsymbol{\Sigma}))\, \mathbf{Q}^\top$ is the eigendecomposition of $\boldsymbol{\Sigma}$. It is therefore clear that the denominator of the r.h.s. is a non-decreasing function of σ_0^2. This implies $|z(\mathbf{x})|$ is a non-increasing function of σ_0^2.

For the limits, it is clear that $\lambda_{\text{min}}(\boldsymbol{\Sigma})$ has limits $1/\lambda_{\text{max}}(\mathbf{H})$ and 0 whenever $\sigma_0^2 \to \infty$ and $\sigma^2 \to 0$, respectively. From these facts, the right limit is immediate from Lemma A.3 while the left limit is directly obtained by noticing that the denominator goes to 1 as $\sigma_0^2 \to 0$. □

Proposition A.4 (Last-layer Laplace)**.** *Let $g : \mathbb{R}^d \to \mathbb{R}$ be a binary linear classifier defined by $g \circ \phi(\mathbf{x}) := \mathbf{w}^\top \phi(\mathbf{x})$ where $\phi : \mathbb{R}^n \to \mathbb{R}^d$ is a ReLU network, modeling a Bernoulli distribution $p(y|\mathbf{x}, \mathbf{w}) = \mathcal{B}(\sigma(g \circ \phi(\mathbf{x})))$ with parameter $\mathbf{w} \in \mathbb{R}^d$. Let $\mathcal{N}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be the posterior obtained via a Laplace approximation with prior $\mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \sigma_0^2\mathbf{I})$ and $\mathbf{H}$ be the Hessian of the negative log-likelihood at $\boldsymbol{\mu}$. Then for any input $\mathbf{x} \in \mathbb{R}^n$, the confidence $\sigma(|z(\mathbf{x})|)$ is a non-increasing function of σ_0^2 with limits*

$$\lim_{\sigma_0^2 \to \infty} \sigma(|z(\mathbf{x})|) \leq \sigma\left(\frac{|\boldsymbol{\mu}^\top \boldsymbol{\phi}|}{1 + \sqrt{\pi/8}\, \lambda_{\text{max}}(\mathbf{H}) \|\boldsymbol{\phi}\|^2}\right)$$
$$\lim_{\sigma_0^2 \to 0} \sigma(|z(\mathbf{x})|) = \sigma(|\boldsymbol{\mu}^\top \boldsymbol{\phi}|) \, .$$

Proof. The assumption on the prior implies that $-\log p(\mathbf{w}) = 1/2\, \mathbf{w}^\top (1/\sigma_0^2 \mathbf{I}) \mathbf{w} + \text{const}$, which has Hessian $1/\sigma_0^2 \mathbf{I}$. Thus, the Hessian of the negative log posterior $-\log p(\mathbf{w}|\mathcal{D}) = -\log p(\mathbf{w}) - \log \prod_{\mathbf{x},t \in \mathcal{D}} p(y|\mathbf{x}, \mathbf{w})$ is $1/\sigma_0^2 \mathbf{I} + \mathbf{H}$. This implies that the posterior covariance $\boldsymbol{\Sigma}$ of the Laplace approximation is given by

$$\boldsymbol{\Sigma} = \left(\frac{1}{\sigma_0^2}\mathbf{I} + \mathbf{H}\right)^{-1} . \tag{14}$$

Therefore, the ith eigenvalue of $\boldsymbol{\Sigma}$ for any $i = 1, \dots, n$ is

$$\lambda_i(\boldsymbol{\Sigma}) = \frac{1}{1/\sigma_0^2 + \lambda_i(\mathbf{H})} = \frac{\sigma_0^2}{1 + \sigma_0^2 \lambda_i(\mathbf{H})} \, .$$

For all $i = 1, \dots, n$, the derivative of $\lambda_i(\boldsymbol{\Sigma})$ w.r.t. σ_0^2 is $1/(1 + \sigma_0^2 \lambda_i(\mathbf{H}))^2$ which is non-negative. This tells us that $\lambda_i(\boldsymbol{\Sigma})$ is a non-decreasing function of σ_0^2. Furthermore, it is also clear that $\sigma_0^2/(1 + \sigma_0^2\lambda_i(\mathbf{H}))$ goes to $1/\lambda_i(\mathbf{H})$ as σ_0^2 goes to infinity, while it goes to 0 as σ_0^2 goes to zero.

Now, we can write.

$$|z(\mathbf{x})| = \frac{|\boldsymbol{\mu}^\top \boldsymbol{\phi}|}{\sqrt{1 + \pi/8 \sum_{i=1}^d \lambda_i(\boldsymbol{\Sigma}) (\mathbf{Q}^\top \boldsymbol{\phi})_i^2}} \, , \tag{15}$$

where $\boldsymbol{\Sigma} = \mathbf{Q} \operatorname{diag}(\lambda_i(\boldsymbol{\Sigma}), \dots, \lambda_d(\boldsymbol{\Sigma}))\, \mathbf{Q}^\top$ is the eigendecomposition of $\boldsymbol{\Sigma}$. It is therefore clear that the denominator of the r.h.s. is a non-decreasing function of σ_0^2. This implies $|z(\mathbf{x})|$ is a non-increasing function of σ_0^2.

For the limits, it is clear that $\lambda_{\text{min}}(\boldsymbol{\Sigma})$ has limits $1/\lambda_{\text{max}}(\mathbf{H})$ and 0 whenever $\sigma_0^2 \to \infty$ and $\sigma^2 \to 0$, respectively. From these facts, the right limit is immediate from Lemma A.3 while the left limit is directly obtained by noticing that the denominator goes to 1 as $\sigma_0^2 \to 0$. □

B. Laplace Approximations

The theoretical results in the main text essentially tell us that if we have a Gaussian approximate posterior that comes from a Laplace approximation, then using eq. (1) (and eq. (2)) to make predictions can remedy the overconfidence problem on any ReLU network. In this section, we describe

LLLA, DLA, and KFLA: the Laplace methods being used in the main text. For the sake of clarity, we omit biases in the following and revisit the case where biases are included at the end of this section.

B.1. LLLA

In the case of LLLA, we simply perform a Laplace approximation to get the posterior of the weight of the last layer $\mathbf{w}$ while assuming the previous layer to be fixed. I.e. we infer $p(\mathbf{w}|\mathcal{D}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\text{MAP}}, \mathbf{H}^{-1})$ where $\mathbf{H}$ is the Hessian of the negative log-posterior w.r.t. $\mathbf{w}$ at $\mathbf{w}_{\text{MAP}}$. This Hessian could be easily obtained via automatic differentiation. We emphasize that we only deal with the weight at the last layer and not the weight of the whole network, thus the inversion of $\mathbf{H}$ is rarely a problem. For instance, even for large models such as DenseNet-201 (Huang et al., 2017) and ResNet-152 (He et al., 2016) have $d = 1920$ and $d = 2048$ respectively,[7] implying that we only need to do the inversion of a single 1920×1920 or 2048×2048 matrix once.

In the case of multi-class classification, we now have $f : \mathbb{R}^d \to \mathbb{R}^k$ defined by $\boldsymbol{\phi} \mapsto \mathbf{W}_{\text{MAP}}\boldsymbol{\phi}$. We obtain the posterior over a random matrix $\mathbf{W} \in \mathbb{R}^{k \times d}$ in the form $\mathcal{N}(\text{vec}(\mathbf{W})|\text{vec}(\mathbf{W}_{\text{MAP}}), \mathbf{\Sigma})$ for some $\mathbf{\Sigma} \in \mathbb{R}^{dk \times dk}$ SPD. The procedure is still similar to the one described above, since the exact Hessian of the linear multi-class classifier can still be easily and efficiently obtained via automatic differentiation. Note that in this case we need to invert a $dk \times dk$ matrix, which, depending on the size of k, can be quite large.[8]

For a more efficient procedure, we can make a further approximation to the posterior in the multi-class case by assuming the posterior is a matrix Gaussian distribution. We can use the Kronecker-factored Laplace approximation (KFLA) (Ritter et al., 2018), but only for the last layer of the network. That is, we find the Kronecker factorization of the Hessian $\mathbf{H}^{-1} \approx \mathbf{V}^{-1} \otimes \mathbf{U}^{-1}$ via automatic differentiation (Dangel et al., 2020).[9] Then by definition of a matrix Gaussian (Gupta & Nagar, 1999), we immediately obtain the posterior $\mathcal{MN}(\mathbf{W}|\mathbf{W}_{\text{MAP}}, \mathbf{U}, \mathbf{V})$. The distribution of the latent functions is Gaussian, since $\mathbf{f} := \mathbf{W}\boldsymbol{\phi}$ and $p(\mathbf{W}|\mathcal{D}) = \mathcal{MN}(\mathbf{W}|\mathbf{W}_{\text{MAP}}, \mathbf{U}, \mathbf{V})$ imply

$$\begin{aligned} p(\mathbf{f}|\mathcal{D}) &= \mathcal{MN}(\mathbf{f}|\mathbf{W}_{\text{MAP}}\boldsymbol{\phi}, \mathbf{U}, \boldsymbol{\phi}^\top \mathbf{V}\boldsymbol{\phi}) \\ &= \mathcal{N}(\mathbf{f}|\mathbf{W}_{\text{MAP}}\boldsymbol{\phi}, (\boldsymbol{\phi}^\top \mathbf{V}\boldsymbol{\phi}) \otimes \mathbf{U}) \\ &= \mathcal{N}(\mathbf{f}|\mathbf{W}_{\text{MAP}}\boldsymbol{\phi}, (\boldsymbol{\phi}^\top \mathbf{V}\boldsymbol{\phi})\mathbf{U}) , \end{aligned} \tag{16}$$

where the last equality follows since $(\boldsymbol{\phi}^\top \mathbf{V}\boldsymbol{\phi})$ is a scalar. We then have the following integral

$$p(y = i|\mathbf{x}, \mathcal{D}) = \int \text{softmax}(\mathbf{f}, i)\, \mathcal{N}(\mathbf{f}|\mathbf{W}_{\text{MAP}}\boldsymbol{\phi}, (\boldsymbol{\phi}^\top \mathbf{V}\boldsymbol{\phi})\mathbf{U})\, d\mathbf{f} ,$$

which can be approximated via a MC-integral.

While one can always assume that the bias trick is already used, i.e. it is absorbed in the weight matrix/vector, in practice when dealing with pre-trained networks, one does not have such liberty. In this case, one can simply assume that the bias b or $\mathbf{b}$ is independent of the weight $\mathbf{w}$ or $\mathbf{W}$, respectively in the two- and multi-class cases. By using the same Laplace approximation procedure, one can easily get $p(b|\mathcal{D}) := \mathcal{N}(b|\mu_b, \sigma_b^2)$ or $p(\mathbf{b}|\mathcal{D}) := \mathcal{N}(\mathbf{b}|\boldsymbol{\mu}_b, \mathbf{\Sigma}_b)$. This implies $\mathbf{w}^\top\boldsymbol{\phi}+b =: f$ and $\mathbf{W}\boldsymbol{\phi}+\mathbf{b} =: \mathbf{f}$ are also Gaussians given by

$$\mathcal{N}(f|\boldsymbol{\mu}^\top\boldsymbol{\phi} + \mu_b, \boldsymbol{\phi}^\top \mathbf{H}^{-1}\boldsymbol{\phi} + \sigma_b^2) \tag{17}$$

or

$$\mathcal{N}(\mathbf{f}|\mathbf{M}\boldsymbol{\phi} + \mathbf{b}, (\boldsymbol{\phi}^\top \otimes \mathbf{I})\mathbf{\Sigma}(\boldsymbol{\phi} \otimes \mathbf{I}) + \mathbf{\Sigma}_b) , \tag{18}$$

respectively, with $\mathbf{I} \in \mathbb{R}^{k \times k}$ if $\mathbf{W} \in \mathbb{R}^{k \times d}$ and $\boldsymbol{\phi} \in \mathbb{R}^d$. Similarly, in the case when the Kronecker-factored approximation is used, we have

$$p(\mathbf{f}|\mathcal{D}) = \mathcal{N}(\mathbf{f}|\mathbf{W}_{\text{MAP}}\boldsymbol{\phi} + \boldsymbol{\mu}_b, (\boldsymbol{\phi}^\top \mathbf{V}\boldsymbol{\phi})\mathbf{U} + \mathbf{\Sigma}_b) . \tag{19}$$

We present the pseudocodes of LLLA in Algorithms 1 and 2.

Algorithm 1 LLLA with exact Hessian for binary classification.

```
Input:
    A pre-trained network $f \circ \phi$ with $\mathbf{w}_{\text{MAP}}$ as the weight of $f$, (averaged) cross-entropy loss $\mathcal{L}$, training set $\mathcal{D}_{\text{train}}$, test set $\mathcal{D}_{\text{test}}$, mini-batch size $m$, running average weighting $\rho$, and prior precision $\tau_0 = 1/\sigma_0^2$.
Output:
    Predictions $\mathcal{P}$ containing $p(y = 1|\mathbf{x}, \mathcal{D}_{\text{train}})\ \forall \mathbf{x} \in \mathcal{D}_{\text{test}}$.
$\mathbf{\Lambda} = 0 \in \mathbb{R}^{d \times d}$
for $i = 1, \dots, |\mathcal{D}_{\text{train}}|/m$ do
    $\mathbf{X}_i, \mathbf{y}_i = \text{sampleMinibatch}(\mathcal{D}_{\text{train}}, m)$
    $\mathbf{A}_i, \mathbf{B}_i = \text{getHessian}(\mathcal{L}(f \circ \phi(\mathbf{X}_i), \mathbf{y}_i), \mathbf{w}_{\text{MAP}})$
    $\mathbf{\Lambda} = \rho\mathbf{\Lambda} + (1 - \rho)\mathbf{\Lambda}_i$
end for
$\mathbf{\Sigma} = (|\mathcal{D}_{\text{train}}|\, \mathbf{\Lambda} + \tau_0\, \mathbf{I})^{-1}$
$p(\mathbf{w}|\mathcal{D}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\text{MAP}}, \mathbf{\Sigma})$
$\mathcal{Y} = \varnothing$
for all $\mathbf{x} \in \mathcal{D}_{\text{test}}$ do
    $y = \sigma(\mathbf{w}_{\text{MAP}}^\top\boldsymbol{\phi}/(1 + \pi/8\, \boldsymbol{\phi}^\top \mathbf{\Sigma}\boldsymbol{\phi})^{1/2})$
    $\mathcal{Y} = \mathcal{P} \cup \{y\}$
end for
```

[7] Based on the implementations available in the TorchVision package.

[8] For example, the ImageNet dataset has $k = 1000$.

[9] In practice, we take the running average of the Kronecker factors of the Hessian over the mini-batches.

Algorithm 2 LLLA with Kronecker-factored Hessian for multi-class classification.

```
Input:
    A pre-trained network $f \circ \phi$ with $\mathbf{W}_\text{MAP}$ as the weight of $f$, (averaged) cross-entropy loss $\mathcal{L}$, training set $\mathcal{D}_\text{train}$, test set $\mathcal{D}_\text{test}$, mini-batch size $m$, number of samples $s$, running average weighting $\rho$, and prior precision $\tau_0 = 1/\sigma_0^2$.
Output:
    Predictions $\mathcal{P}$ containing $p(y = i|\mathbf{x}, \mathcal{D}_\text{train}) \, \forall \mathbf{x} \in \mathcal{D}_\text{test} \, \forall i \in \{1, \dots, k\}$.
$\mathbf{A} = 0 \in \mathbb{R}^{k \times k}, \mathbf{B} = 0 \in \mathbb{R}^{d \times d}$
for $i = 1, \dots, |\mathcal{D}_\text{train}|/m$ do
    $\mathbf{X}_i, \mathbf{y}_i = \text{sampleMinibatch}(\mathcal{D}_\text{train}, m)$
    $\mathbf{A}_i, \mathbf{B}_i = \text{KronFactors}(\mathcal{L}(f \circ \phi(\mathbf{X}_i), \mathbf{y}_i), \mathbf{W}_\text{MAP})$
    $\mathbf{A} = \rho \mathbf{A} + (1 - \rho)\mathbf{A}_i$
    $\mathbf{B} = \rho \mathbf{B} + (1 - \rho)\mathbf{B}_i$
end for
$\mathbf{U} = (\sqrt{|\mathcal{D}_\text{train}|}\, \mathbf{A} + \sqrt{\tau_0}\, \mathbf{I})^{-1}$
$\mathbf{V} = (\sqrt{|\mathcal{D}_\text{train}|}\, \mathbf{B} + \sqrt{\tau_0}\, \mathbf{I})^{-1}$
$p(\mathbf{W}|\mathcal{D}) = \mathcal{MN}(\mathbf{W}|\mathbf{W}_\text{MAP}, \mathbf{U}, \mathbf{V})$
$\mathcal{Y} = \varnothing$
for all $\mathbf{x} \in \mathcal{D}_\text{test}$ do
    $p(\mathbf{f}|\mathcal{D}) = \mathcal{N}(\mathbf{f}|\mathbf{W}_\text{MAP}\boldsymbol{\phi}, (\boldsymbol{\phi}^\top \mathbf{V} \boldsymbol{\phi})\mathbf{U})$
    $\mathbf{y} = \mathbf{0}$
    for $j = 1, \dots, s$ do
        $\mathbf{f}_j \sim p(\mathbf{f}|\mathcal{D})$
        $\mathbf{y} = \mathbf{y} + \text{softmax}(\mathbf{f}_j)$
    end for
    $\mathbf{y} = \mathbf{y}/m$
    $\mathcal{Y} = \mathcal{Y} \cup \{\mathbf{y}\}$
end for
```

B.2. DLA

In this method, we aim at inferring the *diagonal* of the covariance of the Gaussian over the whole layer of a network. Instead of using the exact diagonal Hessian, we use the diagonal of the Fisher information matrix $\mathbf{F}$ of the network (Ritter et al., 2018) as follows

$$\begin{aligned}\text{diag}(\boldsymbol{\Sigma}) &\approx (\sigma_0^2 + \text{diag}(\mathbf{F}))^{-1} \\ &= (\sigma_0^2 + \mathbb{E}_{\mathbf{y} \sim p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}), \mathbf{x} \sim \mathcal{D}}(\nabla_{\boldsymbol{\theta}}\, p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}))^2)^{-1} \,.\end{aligned}$$

Thus, one simply needs to do several backpropagation to compute the gradients of all weight matrices of the network. This gives rise to the Gaussian posterior $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}_\text{MAP}, \text{diag}(\boldsymbol{\Sigma}))$. During prediction, an MC-integration scheme is employed: we repeatedly sample a whole network and average their predictions. That is, for each layer $l \in \{1, \dots, L\}$, we sample the lth layer's weight matrix $\mathbf{W}^l \sim \mathcal{N}(\mathbf{W}^l|\mathbf{W}^l_\text{MAP}, \text{diag}(\boldsymbol{\Sigma}))$ by computing

$$\begin{aligned}\mathbf{e} &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ \text{vec}(\mathbf{W}^l) &= \text{vec}(\mathbf{W}^l_\text{MAP} + \mathbf{e} \odot \text{diag}(\boldsymbol{\Sigma}_l)^{\frac{1}{2}}) \,,\end{aligned}$$

where we have denoted the covariance matrix of the lth layer as $\boldsymbol{\Sigma}_l$. Note that, the computational cost for doing prediction scales with the size of the network, thus this scheme is already orders of magnitude more expensive than LLLA, cf. Table 3.

B.3. KFLA

LLLA with a matrix normal distribution as described in the previous section is a special case of KFLA (Ritter et al., 2018). In KFLA, similar to DLA, we aim to infer the posterior of the whole network parameters and not just those of the last layer. Concretely, for each layer $l \in \{1, \dots, L\}$, we infer the posterior

$$\begin{aligned}p(\mathbf{W}^l|\mathcal{D}) &\approx \mathcal{MN}(\mathbf{W}^l|\mathbf{W}^l_\text{MAP}, \mathbf{U}^l, \mathbf{V}^l) \\ &= \mathcal{N}(\text{vec}(\mathbf{W}^l)|\text{vec}(\mathbf{W}^l_\text{MAP}), \mathbf{V}^l \otimes \mathbf{U}^l) \,,\end{aligned}$$

where

$$\begin{aligned}\mathbf{U}^l &= (\sqrt{|\mathcal{D}|}\mathbf{A} + 1/\sigma_0^2 \mathbf{I})^{-1} \,, \\ \mathbf{V}^l &= (\sqrt{|\mathcal{D}|}\mathbf{B} + 1/\sigma_0^2 \mathbf{I})^{-1} \,,\end{aligned}$$

and $\mathbf{A}, \mathbf{B}$ are the Kronecker-factors—e.g. obtained KFAC (Martens & Grosse, 2015)—of the Hessian of the loss w.r.t. $\mathbf{W}^l$.

During predictions, as in DLA, we also use MC-integration to compute the posterior predictive distribution. That is, at each layer $l \in \{1, \dots, L\}$, we sample the lth layer's weight matrix $\mathbf{W}^l$ via

$$\begin{aligned}\mathbf{E} &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ \mathbf{W}^l &= \mathbf{W}^l_\text{MAP} + (\mathbf{U}^l)^{\frac{1}{2}} \mathbf{E} (\mathbf{V}^l)^{\frac{1}{2}} \,,\end{aligned}$$

where $\mathbf{S}, \mathbf{T}$ are the Cholesky factors such that $(\mathbf{U}^l)^{\frac{1}{2}}(\mathbf{U}^{\frac{1}{2}})^\top = \mathbf{U}$ and $(\mathbf{V}^{\frac{1}{2}})^\top \mathbf{V}^{\frac{1}{2}} = \mathbf{V}$. Again, the cost for doing prediction scales with the size of the network, and it is clear that KFLA is more expensive than DLA.

C. Training Detail

We train all networks we use in Table 2 for 100 epochs with batch size of 128. We use ADAM and SGD with 0.9 momentum with the initial learning rates of 0.001 and 0.1 for MNIST and CIFAR-10/SVHN/CIFAR1-00 experiments, respectively, and we divide them by 10 at epoch 50, 75, and 95. Standard data augmentations, i.e. random crop and standardization are also used for training the network on CIFAR-10. We use a graphic card with 11GB memory for all computation.

D. Further Experiments

D.1. Non-Bayesian Baselines

To represent non-Bayesian Gaussian approximations, we use the following simple baseline: Given a ReLU network, we assume that the distribution over the last-layer's weights is an isotropic Gaussian $\mathcal{N}(\mathbf{0}, \sigma_0^2 \mathbf{I})$, where σ_0^2 is found via cross-validation, optimizing (8). The results on toy datasets are presented in Figure 6. We found that our theoretical analysis hold under this setup, in the sense that far-away from the training data, the confidence is constant less than one. However, predictions around the training data lack structure, unlike the predictions of Bayesian methods. This is because an isotropic Gaussian is too simple and does not capture the structure of the training data. In contrast, Bayesian methods, in particular Laplace approximations, capture this structure in the Hessian of the negative log-likelihood.

D.2. Histograms

To give a more fine-grained perspective of the results in Tables 1 and 2, we show the histograms in Figures 9 and 10. The histograms of both the in-distribution data and far-away OOD data are close together in both MAP and temperature scaling methods, leading to low AUR scores. Meanwhile LLLA (representing Bayesian methods) yields clear separations.

D.3. Asymptotic Confidence of Multi-class Problems

In Figure 7, we present the multi-class counterpart of Figure 5. We found that, as in the binary case, the Bayesian method (LLLA) mitigates overconfidence in the asymptotic regime. We observed, however, that LLLA is less effective in MNIST, which might be due to the architecture choice and the training procedure used: The eigenvalues of the Gaussian posterior's covariance might be too small such that (4) is still large.

D.4. Rotated MNIST

Following Ovadia et al. (2019), we further benchmark the methods in Section 4.3 on the rotated MNIST dataset. The goal is to see whether Bayesian methods could detect dataset shifts of increasing strength in term of Brier score (Brier, 1950). Note that lower Brier score is favorable. We present the results in Figure 8. We found that all Bayesian methods achieve lower Brier score compared to MAP and temperature scaling, signifying that Bayesian methods are better at detecting dataset shift.

D.5. Adversarial Examples

The adversarial datasets ("Adversarial" and "FarAwayAdv", cf. Table 2) are constructed as follows. For "Adversarial": We use the standard PGD attack (Madry et al., 2018) on a uniform noise dataset of size 2000. The objective is to maximize the confidence of the MAP model (resp. ACET and OE below) inside of an ℓ^∞ ball with radius $\epsilon = 0.3$. The optimization is carried out for 40 iterations with a step size of 0.1. We ensure that the resulting adversarial examples are in the image space. For "FarAwayAdv": We use the same construction, but start from the "far-away" Noise datasets as used in Table 2 and we do not project the resulting adversarial examples onto the image space.

D.6. Bayesian Methods on Top of State-of-the-art OOD Detectors

We can also apply all methods we are considering here on top of the state-of-the-art models that are specifically trained to mitigate the overconfidence problem, namely ACET (Hein et al., 2019) and outlier exposure (OE) (Hendrycks et al., 2019). The results are presented in Tables 7 and 8. In general, applying the Bayesian methods improves the models further, especially in the asymptotic regime.

D.7. Frequentist Calibration

Although calibration is a frequentist approach for predictive uncertainty quantification, it is nevertheless interesting to get an insight on whether the properties of the Bayesian predictive distribution lead to a better calibration. To answer this, we use a standard metric (Naeini et al., 2015; Guo et al., 2017): the expected calibration error (ECE). We use the same models along with the same hyperparameters as we have used in the previous OOD experiments. We present the results in Table 5. We found that all the Bayesian methods are competitive to the temperature scaling method, which is specifically constructed for improving the frequentist calibration.

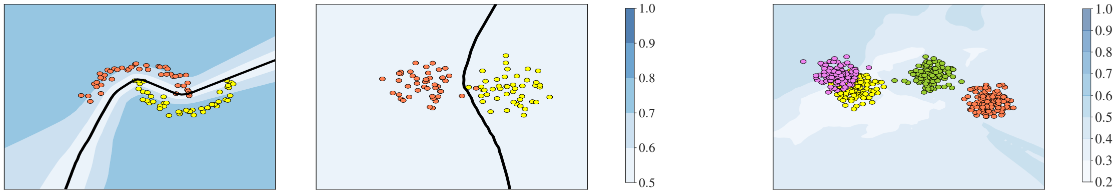
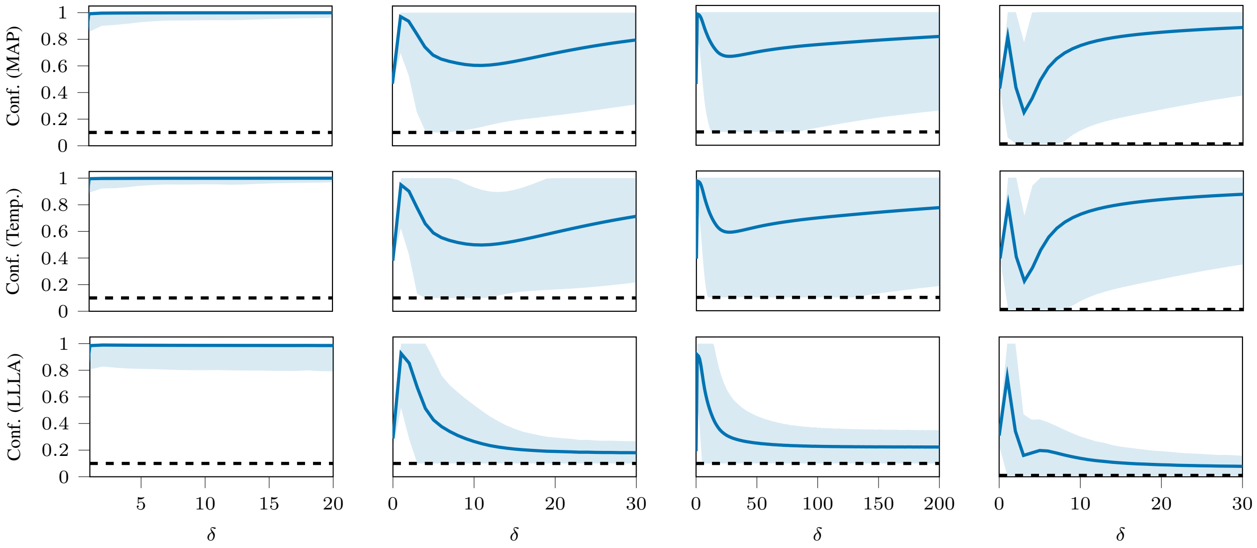


Figure 6. Binary and multi-class toy classification results using a simple isotropic Gaussian baseline. Black lines represent decision boundary, shades represent confidence.



(a) MNIST

(b) CIFAR10

(c) SVHN

(d) CIFAR100

Figure 7. The multi-class confidence of MAP (top row), temperature scaling (middle row), and LLLA (bottom row) as functions of δ over the test sets of the multi-class datasets. Thick blue lines and shades correspond to means and ±3 standard deviations. Dotted lines signify the desirable confidence for δ sufficiently high.

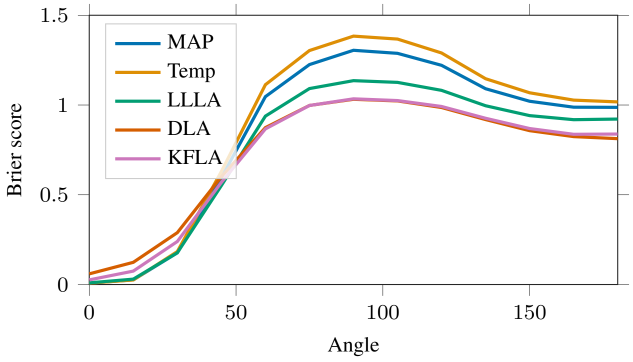
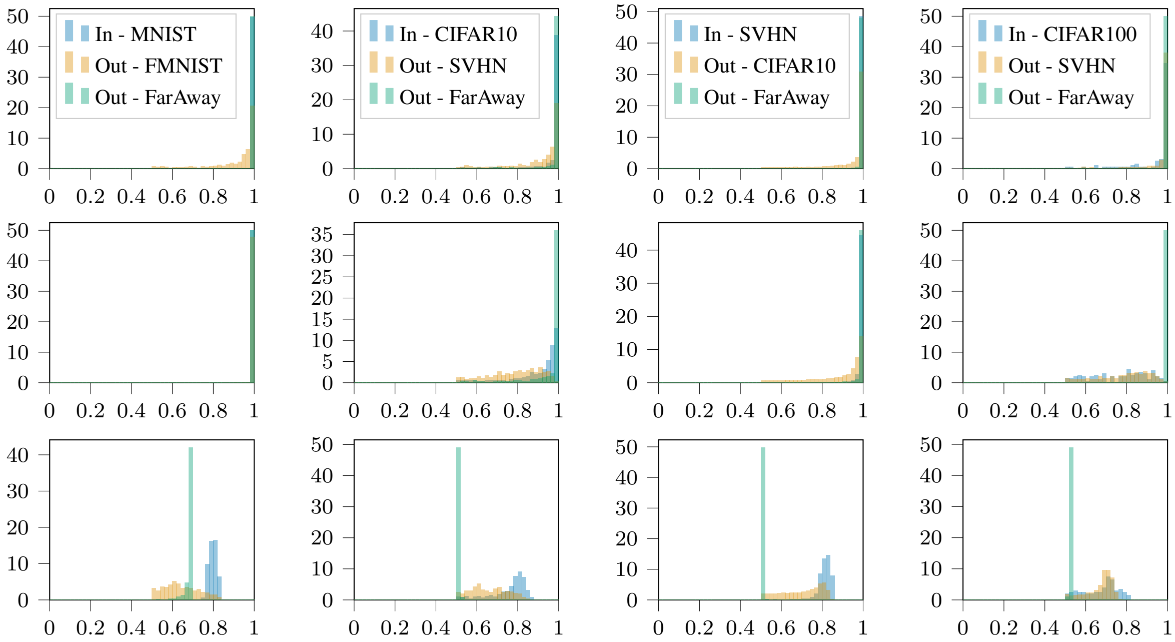


Figure 8. Brier scores (lower is better) over the rotated MNIST dataset. Values shown are means over ten trials. The standard deviations are very small and not visually observable.

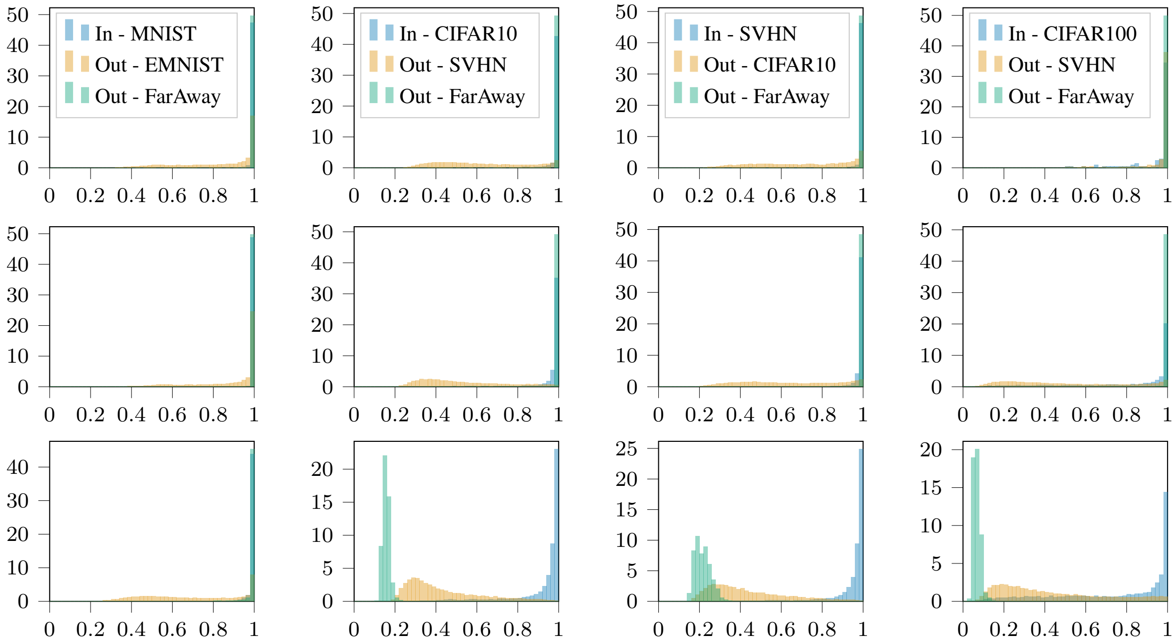
Table 5. Expected calibration errors (ECE).

	MNIST	**CIFAR10**	**SVHN**	**CIFAR100**
MAP	**6.7±0.3**	13.1±0.2	10.1±0.2	8.1±0.3
+Temp.	11.4±2.2	**3.6±0.6**	**2.1±0.5**	6.4±0.5
+LLLA	6.9±0.3	**3.6±0.6**	5.2±0.8	4.8±0.3
+DLA	15.5±0.2	6.9±0.1	8.3±0.0	**4.7±0.3**
+KFLA	9.7±0.3	7.9±0.1	6.5±0.1	5.6±0.4
ACET	**5.9±0.2**	15.8±0.4	11.9±0.2	10.1±0.4
+Temp.	11.0±1.5	**3.7±0.8**	2.3±0.4	6.4±0.4
+LLLA	6.1±0.2	12.3±0.7	9.3±0.5	6.9±0.3
+DLA	6.2±0.3	4.3±0.3	**2.0±0.1**	6.0±0.3
+KFLA	6.1±0.3	4.3±0.2	2.1±0.1	**4.6±0.2**
OE	14.7±1.2	15.8±0.3	11.0±0.1	25.0±0.2
+Temp.	9.0±2.3	23.3±0.7	**3.7±0.7**	**19.4±0.2**
+LLLA	**6.5±0.6**	**14.6±0.2**	4.1±0.3	24.9±0.4
+DLA	9.1±0.6	15.8±0.3	7.2±0.1	29.0±0.2
+KFLA	10.1±0.9	15.9±0.3	6.4±0.1	29.0±0.2



(a) Bin.-MNIST (b) Bin.-CIFAR10 (c) Bin.-SVHN (d) Bin.-CIFAR100

Figure 9. The histograms of MAP (top row), temperature scaling (middle row), and LLLA (bottom row) over the binary datasets. Each entry “Out - FarAway” refers to the OOD dataset obtained by scaling the corresponding in-distribution dataset with some $\delta > 0$.



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

Figure 10. The histograms of MAP (top row), temperature scaling (middle row), and LLLA (bottom row) over the multi-class datasets. Each entry “Out - FarAway” refers to the OOD dataset obtained by scaling the corresponding in-distribution dataset with some $\delta > 0$.

Table 6. Adversarial OOD detection results.

	MAP		**+Temp.**		**+LLLA**		**+DLA**		**+KFLA**	
	MMC	AUR	MMC	AUR	MMC	AUR	MMC	AUR	MMC	AUR
MNIST - Adversarial	100.0±0.0	0.3±0.0	100.0±0.0	6.8±4.1	100.0±0.0	5.3±0.1	99.6±0.2	2.0±0.9	**91.3±1.2**	**69.2±3.5**
MNIST - FarAwayAdv	100.0±0.0	0.1±0.0	100.0±0.0	6.8±4.1	99.9±0.0	9.3±0.6	85.3±1.4	53.0±3.8	**55.6±2.0**	**97.4±0.3**
CIFAR10 - Adversarial	100.0±0.0	0.0±0.0	100.0±0.0	0.0±0.0	99.7±0.0	**9.1±0.1**	99.3±0.1	9.0±1.0	**99.2±0.0**	5.8±0.4
CIFAR10 - FarAwayAdv	99.5±0.0	8.8±0.0	99.2±0.0	7.9±0.1	**17.4±0.1**	**100.0±0.0**	61.3±2.4	89.4±1.0	61.2±1.3	87.8±0.8
SVHN - Adversarial	100.0±0.0	0.0±0.0	100.0±0.0	0.0±0.0	**97.6±0.0**	**32.5±0.3**	98.6±0.0	6.8±0.3	98.6±0.1	9.6±0.4
SVHN - FarAwayAdv	99.7±0.0	7.7±0.0	99.5±0.0	6.9±0.1	**27.5±0.1**	**99.6±0.0**	61.7±1.4	92.4±0.9	61.0±1.2	94.4±0.3
CIFAR100 - Adversarial	**100.0±0.0**	0.0±0.0	**100.0±0.0**	0.0±0.0	**100.0±0.0**	**0.2±0.0**	**100.0±0.0**	0.1±0.0	**100.0±0.0**	0.0±0.0
CIFAR100 - FarAwayAdv	100.0±0.0	1.3±0.0	99.9±0.0	1.2±0.0	**5.9±0.0**	**99.9±0.0**	42.0±1.5	83.9±0.9	42.3±1.8	80.8±1.2

Table 7. OOD detection results when applying post-hoc Bayesian methods on top of models trained with ACET (Hein et al., 2019).

	MAP		**+Temp.**		**+LLLA**		**+DLA**		**+KFLA**	
	MMC	AUR	MMC	AUR	MMC	AUR	MMC	AUR	MMC	AUR
MNIST - MNIST	98.9±0.0	-	99.5±0.0	-	98.9±0.0	-	98.9±0.0	-	98.9±0.0	-
MNIST - EMNIST	59.1±0.0	**96.9±0.0**	70.9±1.8	96.5±0.1	**59.0±0.0**	**96.9±0.0**	**59.0±0.0**	**96.9±0.0**	59.1±0.0	**96.9±0.0**
MNIST - FMNIST	**10.2±0.0**	**100.0±0.0**	10.3±0.0	**100.0±0.0**	**10.2±0.0**	**100.0±0.0**	**10.2±0.0**	**100.0±0.0**	**10.2±0.0**	**100.0±0.0**
MNIST - Noise ($\delta = 2000$)	100.0±0.0	0.0±0.0	100.0±0.0	**21.9±9.2**	100.0±0.0	0.3±0.2	**99.9±0.0**	0.3±0.2	100.0±0.0	0.2±0.1
MNIST - Adversarial	**10.0±0.0**	**100.0±0.0**	**10.0±0.0**	**100.0±0.0**	10.1±0.0	**100.0±0.0**	**10.0±0.0**	**100.0±0.0**	**10.0±0.0**	**100.0±0.0**
MNIST - FarAwayAdv	**100.0±0.0**	0.0±0.0	**100.0±0.0**	**21.9±9.2**	**100.0±0.0**	0.1±0.0	**100.0±0.0**	0.2±0.0	**100.0±0.0**	0.1±0.0
CIFAR10 - CIFAR10	97.3±0.0	-	95.2±0.2	-	96.7±0.1	-	94.7±0.0	-	94.9±0.0	-
CIFAR10 - SVHN	62.8±0.0	96.1±0.0	**52.9±0.7**	**96.5±0.0**	59.5±0.5	96.1±0.1	53.1±0.1	96.2±0.1	53.7±0.1	96.2±0.0
CIFAR10 - LSUN	72.1±0.0	92.8±0.0	62.6±0.7	93.2±0.1	68.9±0.6	92.8±0.1	**59.9±0.6**	**93.8±0.2**	60.4±0.2	93.7±0.1
CIFAR10 - Noise ($\delta = 2000$)	100.0±0.0	0.0±0.0	100.0±0.0	0.0±0.0	**16.0±0.0**	**100.0±0.0**	71.7±1.7	92.3±0.7	65.8±1.8	94.4±0.5
CIFAR10 - Adversarial	78.1±0.0	83.1±0.1	71.1±0.5	84.1±0.1	76.8±0.0	83.2±0.1	67.9±0.4	88.5±0.2	**67.7±0.3**	**88.8±0.2**
CIFAR10 - FarAwayAdv	100.0±0.0	0.0±0.0	100.0±0.0	0.0±0.0	**16.0±0.0**	**100.0±0.0**	72.5±2.4	92.1±0.7	70.7±1.9	93.1±0.5
SVHN - SVHN	98.5±0.0	-	97.3±0.2	-	98.3±0.0	-	96.7±0.0	-	96.2±0.0	-
SVHN - CIFAR10	65.9±0.0	95.6±0.0	58.5±0.8	95.7±0.0	64.0±0.3	95.7±0.0	49.8±0.1	**97.5±0.0**	**48.3±0.1**	97.4±0.0
SVHN - LSUN	28.0±0.0	99.3±0.0	24.6±0.3	99.4±0.0	27.8±0.1	99.3±0.0	22.8±0.6	**99.6±0.0**	**21.7±0.6**	**99.6±0.0**
SVHN - Noise ($\delta = 2000$)	17.9±0.2	**100.0±0.0**	16.0±0.3	**100.0±0.0**	**15.0±0.0**	**100.0±0.0**	45.1±2.1	99.0±0.2	41.6±1.3	99.1±0.1
SVHN - Adversarial	10.4±0.0	**100.0±0.0**	**10.3±0.0**	**100.0±0.0**	10.8±0.0	**100.0±0.0**	10.4±0.0	**100.0±0.0**	10.4±0.0	**100.0±0.0**
SVHN - FarAwayAdv	17.6±0.0	**100.0±0.0**	15.7±0.2	**100.0±0.0**	**15.0±0.0**	**100.0±0.0**	44.9±2.6	99.0±0.2	44.8±1.5	98.8±0.1
CIFAR100 - CIFAR100	82.0±0.1	-	78.1±0.5	-	79.6±0.1	-	78.7±0.1	-	76.3±0.1	-
CIFAR100 - SVHN	57.1±0.0	77.8±0.1	**49.5±0.8**	**78.7±0.1**	52.7±0.1	78.4±0.1	52.4±0.0	77.7±0.1	**49.5±0.0**	77.4±0.2
CIFAR100 - LSUN	55.1±0.0	78.8±0.1	48.3±0.7	79.0±0.1	50.6±0.1	**79.5±0.1**	49.8±0.1	79.3±0.1	**46.8±0.2**	79.2±0.2
CIFAR100 - Noise ($\delta = 2000$)	99.3±0.1	4.2±0.2	99.2±0.1	3.8±0.2	**5.4±0.0**	**100.0±0.0**	58.5±1.3	76.1±0.9	51.8±1.1	77.6±0.9
CIFAR100 - Adversarial	1.5±0.0	**100.0±0.0**	**1.4±0.0**	**100.0±0.0**	1.5±0.0	**100.0±0.0**	**1.4±0.0**	**100.0±0.0**	**1.4±0.0**	**100.0±0.0**
CIFAR100 - FarAwayAdv	99.7±0.0	3.4±0.0	99.6±0.0	3.1±0.0	**5.4±0.0**	**100.0±0.0**	57.7±1.5	76.6±1.0	57.9±1.4	73.4±1.0

Table 8. OOD detection results when applying post-hoc Bayesian methods on top of models trained with outlier exposure (OE) (Hendrycks et al., 2019).

	MAP		**+Temp.**		**+LLLA**		**+DLA**		**+KFLA**	
	MMC	AUR	MMC	AUR	MMC	AUR	MMC	AUR	MMC	AUR
MNIST - MNIST	99.6±0.0	-	99.4±0.1	-	97.8±0.8	-	99.4±0.0	-	99.4±0.0	-
MNIST - EMNIST	84.2±0.0	96.0±0.1	77.1±2.6	**96.3±0.1**	**67.3±3.1**	94.3±0.7	79.6±0.0	95.6±0.1	79.1±0.0	95.9±0.1
MNIST - FMNIST	27.9±0.0	**99.9±0.0**	**22.8±1.5**	**99.9±0.0**	25.6±1.6	**99.9±0.0**	27.5±0.1	**99.9±0.0**	27.3±0.0	**99.9±0.0**
MNIST - Noise ($\delta = 2000$)	99.9±0.0	26.4±0.2	99.9±0.0	5.0±2.4	66.0±0.6	95.9±0.4	58.4±0.3	97.6±0.3	**49.8±0.3**	**99.3±0.1**
MNIST - Adversarial	40.5±0.0	98.8±0.0	**35.2±1.1**	99.1±0.0	38.7±0.0	98.1±0.0	38.1±0.0	98.7±0.0	35.8±0.1	**99.2±0.0**
MNIST - FarAwayAdv	100.0±0.0	25.5±0.2	100.0±0.0	3.6±2.4	66.6±0.1	95.7±0.1	59.2±0.3	97.2±0.2	**50.5±0.3**	**99.3±0.1**
CIFAR10 - CIFAR10	89.4±0.1	-	92.5±0.4	-	89.2±0.1	-	89.3±0.1	-	89.3±0.1	-
CIFAR10 - SVHN	**10.8±0.0**	**98.8±0.0**	11.2±0.1	**98.8±0.0**	10.9±0.0	98.7±0.0	**10.8±0.0**	**98.8±0.0**	**10.8±0.0**	**98.8±0.0**
CIFAR10 - LSUN	**10.4±0.0**	**98.6±0.0**	10.7±0.1	**98.6±0.0**	10.6±0.0	98.5±0.1	**10.4±0.0**	**98.6±0.0**	**10.4±0.0**	**98.6±0.0**
CIFAR10 - Noise ($\delta = 2000$)	99.1±0.1	6.5±0.6	99.4±0.1	7.6±0.7	**25.0±0.1**	**93.6±0.1**	77.9±1.0	79.5±2.0	72.7±1.5	84.6±1.2
CIFAR10 - Adversarial	**98.5±0.0**	2.4±0.0	98.8±0.0	**2.6±0.2**	**98.5±0.0**	2.4±0.0	**98.5±0.0**	2.4±0.0	**98.5±0.0**	2.4±0.0
CIFAR10 - FarAwayAdv	99.5±0.0	5.2±0.0	99.8±0.0	6.2±0.3	**25.1±0.1**	**93.6±0.1**	79.4±1.1	78.4±1.9	78.1±1.4	79.6±1.7
SVHN - SVHN	97.4±0.0	-	95.8±0.3	-	95.7±0.2	-	92.5±0.0	-	93.5±0.0	-
SVHN - CIFAR10	10.2±0.0	**100.0±0.0**	**10.1±0.0**	**100.0±0.0**	14.3±0.6	99.9±0.0	10.8±0.0	**100.0±0.0**	10.8±0.0	**100.0±0.0**
SVHN - LSUN	**10.1±0.0**	**100.0±0.0**	**10.1±0.0**	**100.0±0.0**	14.2±0.6	99.9±0.0	10.8±0.0	**100.0±0.0**	10.9±0.1	**100.0±0.0**
SVHN - Noise ($\delta = 2000$)	99.7±0.0	3.0±0.2	99.6±0.1	2.7±0.2	**16.2±0.0**	**99.7±0.0**	31.5±1.4	98.4±0.2	33.0±1.3	98.4±0.2
SVHN - Adversarial	44.9±0.0	98.2±0.0	34.4±0.7	98.5±0.0	34.2±0.0	98.5±0.0	**17.9±0.2**	99.5±0.0	18.2±0.2	**99.6±0.0**
SVHN - FarAwayAdv	99.9±0.0	2.4±0.0	99.8±0.0	2.2±0.0	**16.3±0.0**	**99.7±0.0**	32.1±1.2	98.3±0.2	31.7±1.6	98.6±0.2
CIFAR100 - CIFAR100	59.6±0.2	-	71.8±0.5	-	54.9±0.2	-	51.5±0.2	-	52.0±0.2	-
CIFAR100 - SVHN	**3.6±0.0**	**93.5±0.1**	7.2±0.2	93.4±0.1	**3.6±0.2**	92.9±0.5	**3.6±0.0**	93.2±0.1	**3.6±0.0**	93.4±0.1
CIFAR100 - LSUN	2.6±0.0	95.4±0.1	5.0±0.1	95.3±0.1	2.9±0.1	94.6±0.2	**2.5±0.1**	**95.9±0.2**	**2.5±0.1**	**95.9±0.2**
CIFAR100 - Noise ($\delta = 2000$)	100.0±0.0	1.3±0.0	100.0±0.0	7.3±0.7	**25.3±0.1**	**64.5±0.2**	89.7±1.9	25.0±2.7	82.4±1.4	33.5±1.3
CIFAR100 - Adversarial	95.6±0.0	21.7±0.1	96.7±0.0	24.6±0.4	**67.3±0.1**	**40.2±0.2**	89.8±0.3	23.9±0.3	89.8±0.2	24.9±0.2
CIFAR100 - FarAwayAdv	100.0±0.0	1.3±0.0	100.0±0.0	7.3±0.7	**25.3±0.1**	**64.5±0.2**	89.4±1.6	25.6±2.2	89.1±1.9	27.2±2.2