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 $\mathbf{x} \in \mathbb{R}^n$,*

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

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

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

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 $\|\mathbf{A}\mathbf{z}\|^2 \geq s_{\min}^2(\mathbf{A}) \|\mathbf{z}\|^2$.*

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,

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

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 $s_{\min}(\mathbf{J}) = s_{\min}(\mathbf{J}^\top)$, we conclude that

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

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:

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

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 $\mathbf{x} \in \mathbb{R}^n$,*

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

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

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

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,

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

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

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

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:

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

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 $i = 1, \ldots, n$ 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 $i = 1, \ldots, n$, 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 $\boldsymbol{\Sigma} = \mathbf{Q}\,\text{diag}(\lambda_i(\boldsymbol{\Sigma}), \ldots, \lambda_d(\boldsymbol{\Sigma}))\,\mathbf{Q}^\top$ 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 $i = 1, \ldots, n$ 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 $i = 1, \ldots, n$, 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 $\boldsymbol{\Sigma} = \mathbf{Q}\,\text{diag}(\lambda_i(\boldsymbol{\Sigma}), \ldots, \lambda_d(\boldsymbol{\Sigma}))\,\mathbf{Q}^\top$ 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 $\mathcal{N}(\text{vec}(\mathbf{W})|\text{vec}(\mathbf{W}_{\text{MAP}}), \boldsymbol{\Sigma})$ for some $\boldsymbol{\Sigma} \in \mathbb{R}^{dk \times dk}$ 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.

[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.

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 $p(\mathbf{b}|\mathcal{D}) := \mathcal{N}(\mathbf{b}|\boldsymbol{\mu}_b, \boldsymbol{\Sigma}_b)$. 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

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

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

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

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 ρ, and prior precision $\tau_0 = 1/\sigma_0^2$.
Output:
Predictions $\mathcal{P}$ containing $p(y = 1|\mathbf{x}, \mathcal{D}_{\text{train}})\ \forall \mathbf{x} \in \mathcal{D}_{\text{test}}$.

$\boldsymbol{\Lambda} = 0 \in \mathbb{R}^{d \times d}$
for $i = 1, \ldots, |\mathcal{D}_{\text{train}}|/m$ **do**
  $\mathbf{X}_i, \mathbf{y}_i = \text{sampleMinibatch}(\mathcal{D}_{\text{train}}, m)$
  $\mathbf{A}_i, \mathbf{B}_i = \text{getHessian}(\mathcal{L}(f \circ \phi(\mathbf{X}_i), \mathbf{y}_i), \mathbf{w}_{\text{MAP}})$
  $\boldsymbol{\Lambda} = \rho\boldsymbol{\Lambda} + (1 - \rho)\boldsymbol{\Lambda}_i$
end for
$\boldsymbol{\Sigma} = (|\mathcal{D}_{\text{train}}|\, \boldsymbol{\Lambda} + \tau_0\, \mathbf{I})^{-1}$
$p(\mathbf{w}|\mathcal{D}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\text{MAP}}, \boldsymbol{\Sigma})$
$\mathcal{Y} = \varnothing$
for all $\mathbf{x} \in \mathcal{D}_{\text{test}}$ **do**
  $y = \sigma(\mathbf{w}_{\text{MAP}}^\top\boldsymbol{\phi}/(1 + \pi/8\, \boldsymbol{\phi}^\top\boldsymbol{\Sigma}\boldsymbol{\phi})^{1/2})$
  $\mathcal{Y} = \mathcal{P} \cup \{y\}$
end for

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 ρ, 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.

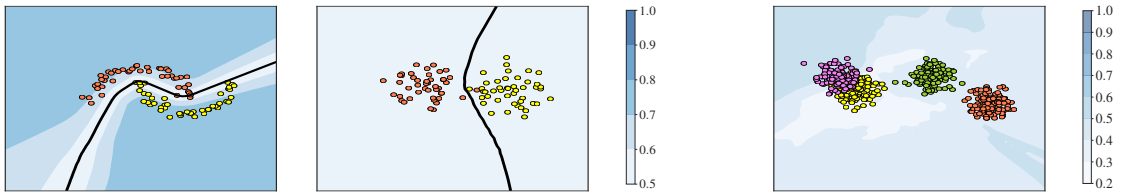
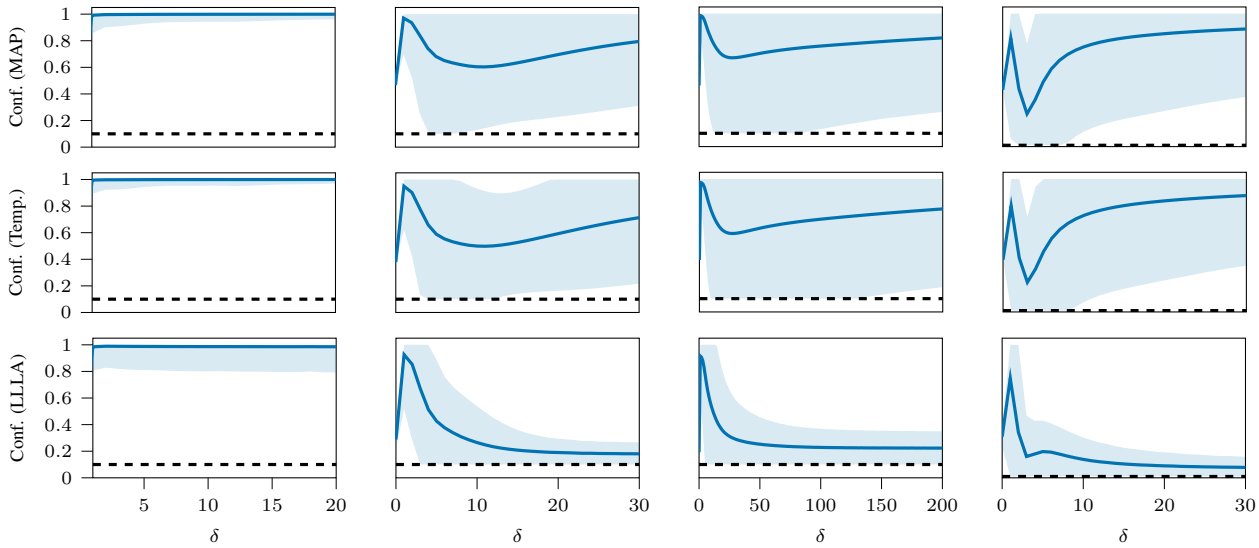


Figure 6. Binary and multi-class toy classification results using a simple isotropic Gaussian baseline. Black lines represent decision boundary, shades represent confidence.



(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.

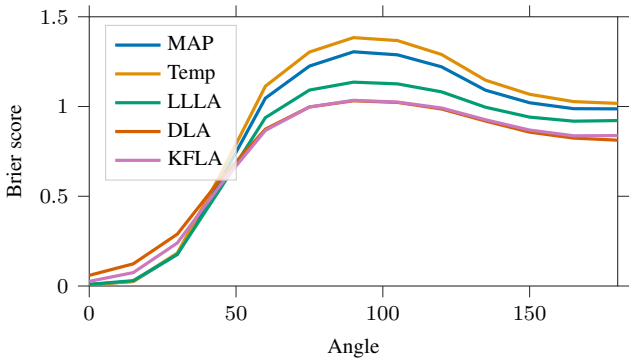


Figure 8. Brier scores (lower is better) over the rotated MNIST dataset. Values shown are means over ten trials. The standard deviations are very small and not visually observable.

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

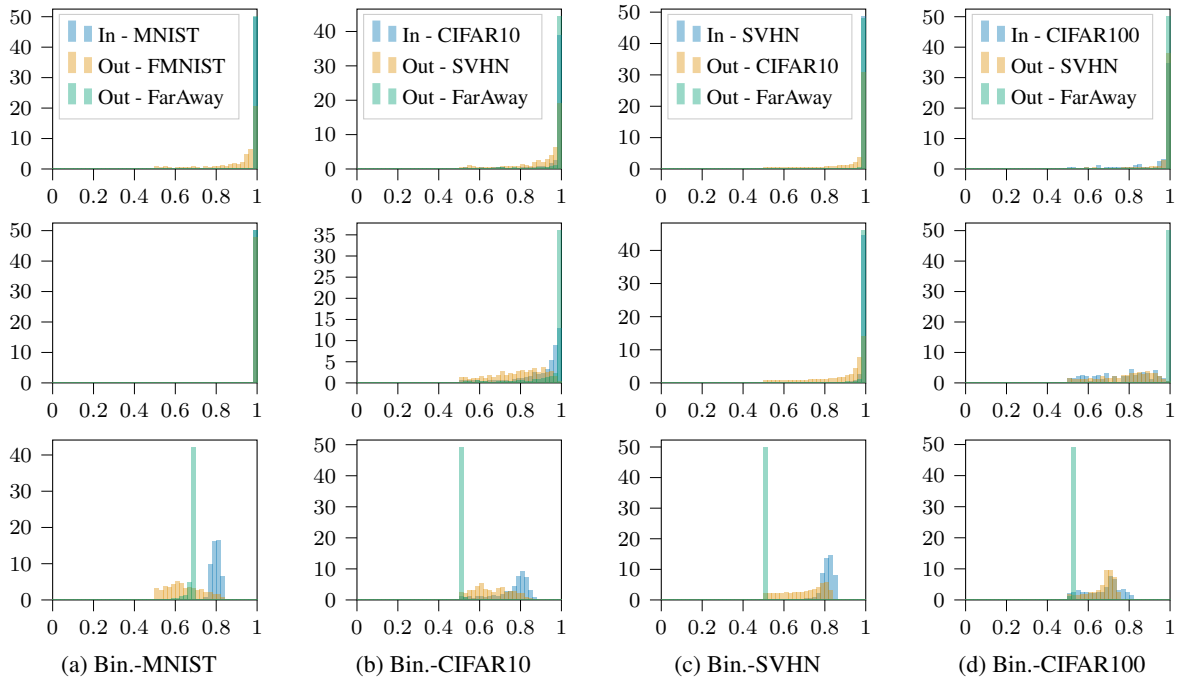


Figure 9. The histograms of MAP (top row), temperature scaling (middle row), and LLLA (bottom row) over the binary datasets. Each entry "Out - FarAway" refers to the OOD dataset obtained by scaling the corresponding in-distribution dataset with some $\delta > 0$.

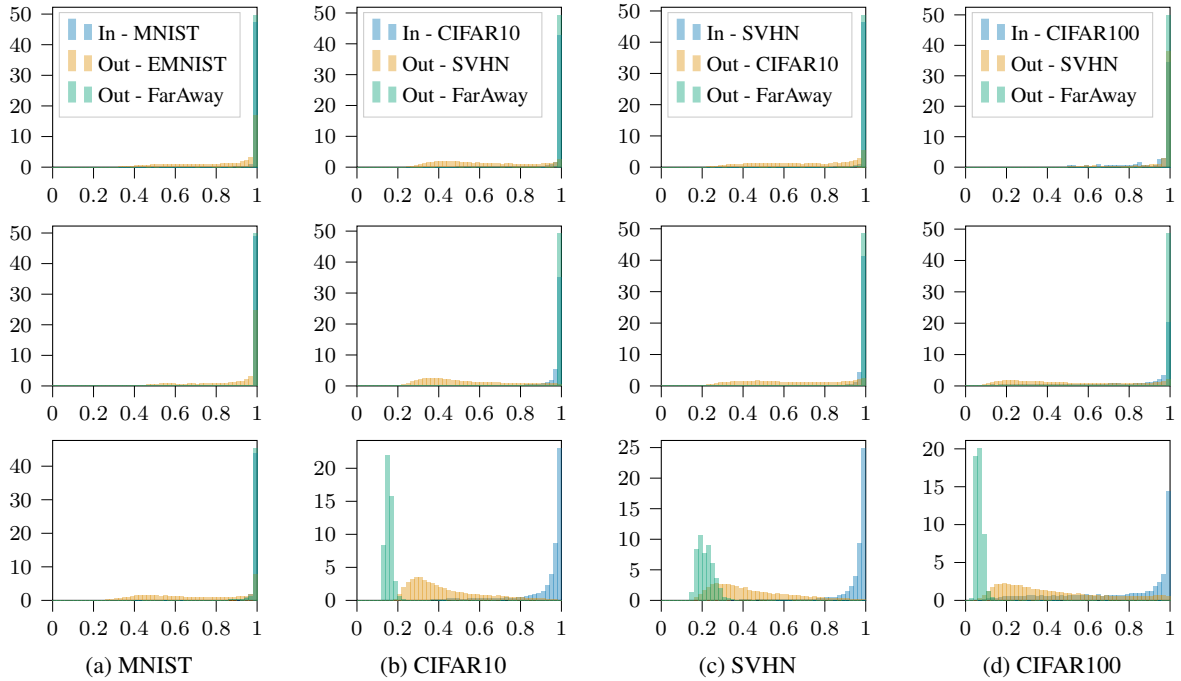


Figure 10. The histograms of MAP (top row), temperature scaling (middle row), and LLLA (bottom row) over the multi-class datasets. Each entry "Out - FarAway" refers to the OOD dataset obtained by scaling the corresponding in-distribution dataset with some $\delta > 0$.

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