Diversity-enhanced Probabilistic Ensemble For Uncertainty Estimation (Supplementary Material)

A PRELIMINARIES

A.1 LAPLACIAN APPROXIMATION

The LA approximates the true posterior of parameters by a Gaussian distribution, i.e., $p(\theta|\mathcal{D},\beta) \approx \mathcal{N}(\theta_{map}, \Sigma)$ where $\Sigma = -(H)^{-1}$ and $H = \nabla^2_\theta \log p(\theta|\mathcal{D},\beta)|_{\theta=\theta_{map}}$.

Efficiently and accurately calculating the Hessian matrix H is the key of LA. Given a standard Gaussian distribution prior $p(\theta|\beta) = \mathcal{N}(\mathbf{0}, \beta^2 I)$ where β is the hyperparameter, we can obtain that

$$\begin{aligned}\nabla^2_\theta \log p(\theta|\mathcal{D},\beta) =& \nabla^2_\theta \log p(\mathcal{D}|\theta) + \nabla^2_\theta \log p(\theta|\beta) \\ =& \sum_{(x,y)\in\mathcal{D}} \nabla^2_\theta \log p(y|x,\theta) + \frac{1}{\beta^2} I\end{aligned} \tag{1}$$

where I is the identity matrix. Basically, computing the second-order derivatives for highly nonlinear neural networks is hard and we leverage the Generalized Gauss-Newton Matrix (GGN) [Schraudolph, 2002] to approximate $\nabla^2_\theta \log p(y|x,\theta)$. Denote the neural network output as $f(x,\theta)$ in general.

$$\begin{aligned}\nabla^2_\theta \log p(y|x,\theta) =& \nabla^2_\theta \log p(y|f(x,\theta)) \\ \approx& J(x) \nabla^2_f p(y|f(x,\theta)) J(x)^T\end{aligned} \tag{2}$$

where $J(x) = \nabla_\theta f(x,\theta)$ is the Jacobian matrix. However, the large matrix multiplication in Eq. (2) may also lead to problems, especially for deep learning models. We use the last-layer Laplacian approximation proposed by Kristiadi et al. [2020], which constructs the posterior approximation only for neural networks' last-layer weights to reduce computational complexity. We use the full Hessian matrix without additional factorization assumptions. To avoid tuning the hyperparameter β, we utilize the marginal likelihood maximization method proposed by Ritter et al. [2018] to do a one-parameter optimization for β. The loss function is the posterior predictive approximated by LA.

$$\beta^* = \arg\max_\beta \sum_{(x,y)\in\mathcal{D}} \log p(y|x,\mathcal{D}) \tag{3}$$

After we compute the Laplacian approximation $\mathcal{N}(\theta_{map}, \Sigma)$, we can perform the Bayesian inference in Eq. (4). Given a new pair of input (x^*, y^*),

$$p(y^*|x^*,\mathcal{D}) = \int p(y^*|x^*,\theta) p(\theta|\mathcal{D}) d\theta \approx \int softmax(f(x,\theta)) \mathcal{N}(\theta; \theta_{map}, \Sigma) d\theta \tag{4}$$

where $softmax(f) = \frac{exp(f)}{\sum_j exp(f_j)}$ is the softmax function and Eq. (4) can be solved either by MC sample average or by probit approximation. Performing the first-order Taylor expansion of $f(x,\theta)$ with respect to θ at θ_{map} yields $f(x,\theta) \approx$

$f(x, \theta_{map}) + J(x)^T(\theta - \theta_{map})$, which indicates that $f(x, \theta) \sim \mathcal{N}(f(x, \theta_{map}), \Sigma^f)$ where $\Sigma^f = J(x)^T \Sigma J(x) \in \mathcal{R}^{C \times C}$. Based on probit approximation,

$$p(y^* = c|x^*, \mathcal{D}) = \frac{exp(\tau^{(c)}(x))}{\sum_j exp(\tau^{(j)}(x))} \text{ where } \tau^{(j)}(x) = \frac{f^{(j)}(x, \theta)}{\sqrt{1 + \frac{\pi}{8}\Sigma^f_{jj}}} \tag{5}$$

where $f^{(j)}(x, \theta) \in \mathcal{R}$ is the jth element of $f(x, \theta)$ and Σ^f_{jj} is the (j, j)th element of Σ^f

A.2 UNCERTAINTY QUANTIFICATION

For classification problems, we estimate the epistemic uncertainty and the aleatoric uncertainty by the mutual information and the expected entropy [Depeweg et al., 2018].

$$\underbrace{\mathcal{H}\left[\mathrm{p}(y|x, \mathcal{D}, \beta)\right]}_{\text{Total Uncertainty}} = \underbrace{\mathcal{I}\left[y, \theta|x, \mathcal{D}, \beta\right]}_{\text{Epistemic Uncertainty}} + \underbrace{\mathbb{E}_{\mathrm{p}(\theta|\mathcal{D},\beta)}\left[\mathcal{H}[\mathrm{p}(y|x, \theta)]\right]}_{\text{Aleatoric Uncertainty}} \tag{6}$$

where $\mathcal{H}$ and $\mathcal{I}$ represent the entropy and mutual information, respectively. More specifically,

$$\begin{aligned} &\mathcal{H}\left[p(y|x, \mathcal{D}, \beta)\right] = \mathcal{H}\left[E_{p(\theta|D,\beta)}[p(y|x, \theta)]\right] \approx \mathcal{H}\left[\frac{1}{S}\sum_{s=1}^{S} p(y|x, \theta^s)\right] \\ &\mathbb{E}_{p(\theta|\mathcal{D},\beta)}\left[\mathcal{H}[p(y|x, \theta)]\right] \approx \frac{1}{S}\sum_{s=1}^{S} \mathcal{H}(p(y|x, \theta^s)) \end{aligned} \tag{7}$$

where $\theta^s \sim p\left(\theta|\mathcal{D}, \beta\right) \approx \sum_{i=1}^{N} \lambda_i \mathcal{N}(\theta; \theta_i, \Sigma_i)$ for the probabilistic ensemble model.

B PROBABILISTIC ENSEMBLE PROPOSITIONS

B.1 PROOF OF PROPOSITION 3.1

Proof. We first introduce the Bernstein-von Mises theorem.

Lemma B.1 (Bernstein-von Mises theorem for Laplacian approximation of the posterior distribution [Gelman, 2011, Kleijn and van der Vaart, 2012])**.** *Under mild regularity conditions (i.e., the likelihood function of θ is continuous, $\sum_{i=1}^{N} \lambda_i \theta_i$ is not on the boundary of the parameter space.), as the sample size $M \to \infty$, the posterior distribution of θ approaches its Laplacian approximation $\mathcal{N}(\theta; \theta_{map}, \Sigma)$. For example,*

$$\sup_{\theta} |p(\theta|\mathcal{D}) - \mathcal{N}(\theta; \theta_{map}, \Sigma))| \to 0 \tag{8}$$

Then for the probabilistic ensemble model, $\theta \sim \sum_{i=1}^{N} \lambda_i \mathcal{N}(\theta; \theta_i, \Sigma_i)$

$$\begin{aligned} &\sup_{\theta} |p(\theta|\mathcal{D}) - \sum_{i=1}^{N} \lambda_i \mathcal{N}(\theta; \theta_i, \Sigma_i)| = \sup_{\theta} |\sum_{i=1}^{N} \lambda_i [p(\theta|\mathcal{D}) - \mathcal{N}(\theta; \theta_i, \Sigma_i)]| \\ &\leq \sum_{i=1}^{N} \lambda_i \sup_{\theta} |p(\theta|\mathcal{D}) - \mathcal{N}(\theta; \theta_i, \Sigma_i)| \to 0 \end{aligned} \tag{9}$$

□

B.2 PROOF OF PROPOSITION 3.2

Proof. The proposed probabilistic ensemble can be an approximate Bayesian method where the Laplacian approximation bridges the connection of randomization-based ensembles and the Bayesian posterior distribution. Given a set of coefficients $\{\lambda_i\}_{i=1}^N$ where $\lambda_i > 0$ and $\sum_{i=1}^N \lambda_i = 1$,

$$p(\theta|\mathcal{D}) = \sum_{i=1}^{N} \lambda_i p(\theta|\mathcal{D}) \approx \sum_{i=1}^{N} \lambda_i \mathcal{N}(\theta; \theta_i, \Sigma_i) \tag{10}$$

Eq. (10) holds since the Laplacian approximation $\mathcal{N}(\theta; \theta_i, \Sigma_i)$ of the ith ensemble model serves independently as an approximation of $p(\theta|\mathcal{D})$. Instead of treating the deep ensemble method as non-Bayesian, we argue that it is necessary to construct the relationship of the deep ensembles with the parameter posterior. The vanilla approximation of posterior for the deep ensemble method can be expressed as $p_{DE}(\theta) = \sum_{i=1}^N \lambda_i \delta(\theta, \theta_i)$ where $\delta(\theta, \theta_i)$ is the delta function that returns 1 if and only if $\theta = \theta_i$ and returns 0 otherwise. Since $p_{DE}(\theta)$ is discrete, however, there might be a big gap between $p_{DE}(\theta)$ and $p(\theta|\mathcal{D})$ when $\theta \notin \{\theta_i\}_{i=1}^N$. For example, the KL divergence between $p(\theta|\mathcal{D})$ and $p_{DE}(\theta)$ is shown in Eq. (11).

$$KL(p(\theta|\mathcal{D})||\sum_{i=1}^{N} \lambda_i \delta(\theta, \theta_i)) = -\mathcal{H}(p(\theta|\mathcal{D})) - \int p(\theta|\mathcal{D}) \log \sum_{i=1}^{N} \lambda_i \delta(\theta, \theta_i) d\theta \tag{11}$$

We can observe that $KL(p(\theta|\mathcal{D})||\sum_{i=1}^N \lambda_i \delta(\theta, \theta_i))$ could be extremely large since $\log \sum_{i=1}^N \lambda_i \delta(\theta, \theta_i) \to -\infty$ when $\theta \notin \{\theta_i\}_{i=1}^N$. It is mainly because the vanilla approximation does not explore the possible values other than the modes. Given a limited number of modes, $p_{DE}(\theta)$ can be used for a Bayesian prediction but is hard to sketch the complex posterior distribution. In contrast, the PE model extends the deep ensemble method for approximate Bayesian inference through exploring each ensemble subspace, enabling a better posterior approximation.

Then, we show that the KL divergence between $p(\theta|D)$ and $\sum_{i=1}^N \lambda_i \mathcal{N}(\theta; \theta_i, \Sigma_i)$ is reduced compared to single-network LA. Based on Jensen's inequality and the convexity of $-\log$, we have that

$$\begin{aligned} KL(p(\theta|\mathcal{D})||\sum_{i=1}^{N} \lambda_i \mathcal{N}(\theta; \theta_i, \Sigma_i)) &= -\mathcal{H}(p(\theta|\mathcal{D})) - \int p(\theta|\mathcal{D}) \log \sum_{i=1}^{N} \lambda_i \mathcal{N}(\theta; \theta_i, \Sigma_i) d\theta \\ &\leq -\mathcal{H}(p(\theta|\mathcal{D})) - \sum_{i=1}^{N} \lambda_i \int p(\theta|\mathcal{D}) \log \mathcal{N}(\theta; \theta_i, \Sigma_i) d\theta \\ &= \sum_{i=1}^{N} \lambda_i KL(p(\theta|\mathcal{D})||\mathcal{N}(\theta; \theta_i, \Sigma_i)) \end{aligned} \tag{12}$$

□

B.3 PROOF OF PROPOSITION 3.3

Proof. At the beginning of the proof, we introduce a lemma based on Liao and Berg [2018].

Lemma B.2 (Sharpening Jensen's inequality [Liao and Berg, 2018]). *Consider a convex function $\phi(\cdot)$ and a scalar random variable $z \in [a, b]$. a, b are finite real numbers. Let $\mu = E[z]$ and denote $r(z)$ as the residual term for the first-order Taylor expansion of $\phi(z)$ at μ, i.e.,*

$$\phi(z) = \phi(\mu) + \phi^{'}(\mu)(z - \mu) + r(z) \tag{13}$$

There must exist a finite number C_{min} such that

$$\mathbb{E}[\phi(z)] - \phi(\mathbb{E}[z]) \geq C_{min} V(z) \tag{14}$$

where $V(z)$ is the variance of z. Especially, $C_{min} \geq \inf_{z \in [a,b]} \frac{\phi^{"}(z)}{2}$ indicates

$$\mathbb{E}[\phi(z)] - \phi(\mathbb{E}[z]) \geq \inf_{z \in [a,b]} \frac{\phi^{"}(z)}{2} V(z) \tag{15}$$

For Proposition 3.3, let $\phi(z) = -\log z$ which is a convex function. Given an input x and the groundtruth label $y \in \{1, 2, ..., C\}$, let $z = p(y|x, \theta) \in [0, 1]$ where θ are the probabilistic ensemble random parameters that follow a mixture of Gaussian distribution. Following lemma B.2, we have the following lower bound for the Jensen's inequality gap.

$$\mathbb{E}_\theta[-\log p(y|x,\theta)] - [-\log \mathbb{E}_\theta[p(y|x,\theta)]] \geq \inf_\theta \frac{1}{2p(y|x,\theta)^2} \mathbb{V}_\theta[p(y|x,\theta)] \tag{16}$$

□

B.4 PROOF OF PROPOSITION 3.4

Proof. First, the vanilla approximation of posterior for the deep ensemble method can be expressed as $p_{DE}(\theta) = \sum_{i=1}^N \lambda_i \delta(\theta, \theta_i)$ where $\delta(\theta, \theta_i)$ is the delta function that returns 1 if and only if $\theta = \theta_i$ and returns 0 otherwise. The mean and variance based on $p_{DE}(\theta)$ are shown in Eq. (17).

$$\begin{aligned} \mu_D &= \mathbb{E}_{\theta \sim p_{DE}(\theta)}[\theta] = \sum_{i=1}^N \lambda_i \theta_i \\ \Sigma_D &= Cov_{\theta \sim p_{DE}(\theta)}[\theta] = \mathbb{E}_{\theta \sim p_{DE}(\theta)}[\theta\theta^T] - \mu\mu^T = \sum_{i=1}^N \lambda_i \theta_i \theta_i^T - \mu_D \mu_D^T \end{aligned} \tag{17}$$

For the probabilistic ensemble $\theta \sim \sum_{i=1}^N \lambda_i \mathcal{N}(\theta; \theta_i, \Sigma_i)$, we have that

$$\begin{aligned} \mu_P &= \mathbb{E}_{\theta \sim p_{PE}(\theta)}[\theta] = \sum_{i=1}^N \lambda_i \theta_i = \mu_D \\ \Sigma_P &= Cov_{\theta \sim p_{PE}(\theta)}[\theta] = \mathbb{E}_{\theta \sim p_{PE}(\theta)}[\theta\theta^T] - \mu\mu^T = \sum_{i=1}^N \lambda_i \mathbb{E}_{\theta \sim \mathcal{N}(\theta; \theta_i, \Sigma_i)}[\theta\theta^T] - \mu_P \mu_P^T \\ &= \sum_{i=1}^N \lambda_i (\theta_i \theta_i^T + \Sigma_i) - \mu_P \mu_P^T = \Sigma_D + \sum_{i=1}^N \lambda_i \Sigma_i \geq \Sigma_D \end{aligned} \tag{18}$$

where $\Sigma_P \geq \Sigma_D$ means $\Sigma_P - \Sigma_D$ is positive semi-definite. Eq. (18) shows that the probabilistic ensemble model has better diversity in terms of variance. □

B.5 PROOF OF PROPOSITION 3.5

Proof. In the beginning, we introduce three lemmas.

Lemma B.3 (From [Hein et al., 2019]. This is also stated in Lemma A.1 in [Kristiadi et al., 2020]). *Denote $\{Q_i\}_{i=1}^R$ be the set of linear regions associated to the ReLU network $f : \mathcal{R}^{|x|} \to \mathcal{R}^C$. For any $x \in \mathcal{R}^{|x|}$, there exists an $\alpha > 0$ and $t \in \{1, 2, ..., R\}$ such that $\delta x \in Q_t$ for all $\delta \geq \alpha$. Furthermore, the restriction of f to Q_t can be written as an affine function $W^T x + q$ for some suitable $W \in \mathcal{R}^{|x| \times C}$ and $q \in \mathcal{R}^C$.*

Lemma B.4 (From Lemma A.2 in [Kristiadi et al., 2020]). *Let $A \in \mathcal{R}^{d_1 \times d_2}$ and $z \in \mathcal{R}^{d_1}$ with $d_1 \geq d_2$, then we have $||Az||^2 \geq s_{min}^2(A)||z||^2$ where $s_{min}(A)$ is the minimum singular value of A.*

Lemma B.5 (From Lemma A.3 in [Kristiadi et al., 2020]). *Let $A \in \mathcal{R}^{d \times d}$ be an SPD matrix and $z \in \mathcal{R}^d$, then we have $z^T Az \geq \lambda_{min}(A)||z||^2$, where $\lambda_{min}(A)$ is the minimum eigenvalue of A.*

Before proving Proposition 3.5, we use the above three lemmas to prove Lemma B.6 first.

Lemma B.6. *Let $f_\theta : R^{|x|} \to R^C$ be a ReLU network for multi-class classification parameterized by θ. Let $|x|$ represent the dimension of x and $\theta \sim \mathcal{N}(\theta; \mu, \Sigma)$ by LA. Then for any input x, the estimated probability based on multi-class probit approximation shown in Eq.* (5) *fulfills*

$$\lim_{\eta \to \infty} |\tau^{(c)}(\delta x)| \leq \frac{||w^{(c)}||}{s_{min}(J^{(c)})\sqrt{\frac{\pi}{8}\lambda_{min}(\Sigma)}} \qquad c = 1, 2, \cdots, C \tag{19}$$

where $w = [w^{(1)}, w^{(2)}, \cdots, w^{(C)}] \in \mathcal{R}^{|x| \times C}$ is a matrix that only depends on μ. $J^{(j)} = \frac{\partial w^{(j)}}{\partial \theta}|_{\theta=\mu}$ is the Jacobian matrix of $w^{(j)}$ with respect to θ at $\theta = \mu$. $\lambda_{min}(\Sigma)$ is the minimum eigenvalue while s_{min} represents the minimum singular value.

Proof. We follow the proof of Theorem 2.3 in [Kristiadi et al., 2020], where they focus on binary classification problems and we extend it to the multi-class cases.

By Lemma B.3, there must exist $\alpha \geq 0$ and a linear region R such that $\delta x \in R$ for all $\delta \geq \alpha$. We have the restriction $f_\theta|_R$ that can be expressed as $f_\theta|_R(x) = w^T x + q$ where $w = [w^{(1)}, w^{(2)}, \cdots, w^{(C)}] \in \mathcal{R}^{|x| \times C}$ and $q \in \mathcal{R}^{|x|}$. w, q can be regarded as constants with respect to δx that only depend on μ. Let $f_\theta(\delta x) = [f_\theta^{(1)}(\delta x), f_\theta^{(2)}(\delta x), ..., f_\theta^{(C)}(\delta x)]^T$ and $q = [q^{(1)}, q^{(2)}, ..., q^{(C)}]^T$. The gradient of $f_\theta^{(c)}(\delta x)(c = 1, 2, ..., C)$ with respect to θ can be expressed as

$$d_c(\delta x) = \frac{\partial \delta w^{(c)^T} x + q^{(c)}}{\partial \theta}|_\mu = \delta(\frac{\partial w^{(c)}}{\partial \theta}|_\mu^T x + \frac{1}{\delta}\frac{\partial q^{(c)}}{\partial \theta}|_\mu) := \delta(J^{(c)^T} x + \frac{1}{\delta}\nabla_\theta q^{(c)}|_\mu) \tag{20}$$

Then based on the multi-class probit approximation shown in Eq. (5), we have

$$\begin{aligned} |\tau^{(c)}(\delta x)| &= \frac{|\delta w^{(c)^T} x + q^{(c)}|}{\sqrt{1 + \frac{\pi}{8} d_c(\delta x)^T \Sigma d_c(\delta x)}} \\ &= \frac{|\delta(w^{(c)^T} x + \frac{1}{\delta} q^{(c)})|}{\sqrt{1 + \frac{\pi}{8}\delta^2 (J^{(c)^T} x + \frac{1}{\delta}\nabla_\theta q^{(c)}|_\mu)^T \Sigma (J^{(c)^T} x + \frac{1}{\delta}\nabla_\theta q^{(c)}|_\mu)}} \\ &= \frac{|w^{(c)^T} x + \frac{1}{\delta} q^{(c)}|}{\sqrt{\frac{1}{\delta^2} + \frac{\pi}{8} (J^{(c)^T} x + \frac{1}{\delta}\nabla_\theta q^{(c)}|_\mu)^T \Sigma (J^{(c)^T} x + \frac{1}{\delta}\nabla_\theta q^{(c)}|_\mu)}} \end{aligned} \tag{21}$$

When $\delta \to \infty$, Eq. (21) becomes

$$\lim_{\delta \to \infty} |\tau_c(\delta x)| = \frac{|w^{(c)^T} x|}{\sqrt{\frac{\pi}{8} (J^{(c)^T} x)^T \Sigma (J^{(c)^T} x)}} \tag{22}$$

Then by using Lemma B.4 and B.5 with Cauchy-Schwarz inequality, and noting that $s_{min}(J^{(c)}) = s_{min}(J^{(c)^T})$, we have

$$\begin{aligned} \lim_{\delta \to \infty} |\tau_c(\delta x)| &= \frac{||w^{(c)^T} x||}{\sqrt{\frac{\pi}{8} (J^{(c)^T} x)^T \Sigma (J^{(c)^T} x)}} \\ &\leq \frac{||w^{(c)}||\ ||x||}{\sqrt{\frac{\pi}{8}\lambda_{min}(\Sigma)||J^{(c)^T} x||^2}} \\ &\leq \frac{||w^{(c)}||\ ||x||}{\sqrt{\frac{\pi}{8}\lambda_{min}(\Sigma) s_{min}^2(J^{(c)^T})||x||^2}} = \frac{||w^{(c)}||}{s_{min}(J^{(c)})\sqrt{\frac{\pi}{8}\lambda_{min}(\Sigma)}} \end{aligned} \tag{23}$$

□

Given a probabilistic ensemble model with N components, let $f_{\theta_i} : R^{|x|} \to R^C$ be a ReLU network for multi-class classification parameterized by θ_i $(i = 1, 2, ..., N)$. For probabilistic ensemble model, $\theta \sim \sum_{i=1}^N \lambda_i \mathcal{N}(\theta; \theta_i, \Sigma_i)$. Based on

Eq. (5) and Lemma B.6, we have the following property for a single model f_{θ_i}.

$$\begin{aligned}\lim_{\delta\to\infty} p_i(y=c|\delta x,\mathcal{D}) &= \frac{\exp(\tau_i^{(c)}(\delta x))}{\sum_{j=1}^{C}\exp(\tau_i^{(j)}(\delta x))} = \frac{1}{1+\sum_{j\neq c}\exp(\tau_i^{(j)}(\delta x)-\tau_i^{(c)}(\delta x))} \\ &\leq \frac{1}{1+\sum_{j\neq c}\exp(-|\tau_i^{(j)}(\delta x)|-|\tau_i^{(c)}(\delta x))|} \\ &\leq \frac{1}{1+\sum_{j\neq c}\exp\{-\frac{||w_i^{(j)}||}{s_{min}(J_i^{(j)})\sqrt{\frac{\pi}{8}\lambda_{min}(\Sigma_i)}}-\frac{||w_i^{(c)}||}{s_{min}(J_i^{(c)})\sqrt{\frac{\pi}{8}\lambda_{min}(\Sigma_i)}}\}}\end{aligned} \tag{24}$$

where $w_i=[w_i^{(1)},w_i^{(2)},\cdots,w_i^{(C)}]\in R^{|x|\times C}$ is a matrix that only depends on θ_i. $J_i^{(j)}=\frac{\partial w_i^{(j)}}{\partial\theta}|_{\theta=\theta_i}$ is the Jacobian matrix of $w_i^{(j)}$ with respect to θ at $\theta=\theta_i$.

Then for the probabilistic ensemble $\theta\sim\sum_{i=1}^{N}\lambda_i\mathcal{N}(\theta;\theta_i,\Sigma_i)$

$$\begin{aligned}\lim_{\delta\to\infty} p_{PE}(y=c|\delta x,\mathcal{D}) &= \lim_{\delta\to\infty}\int p(y=c|\delta x,\theta)\sum_{i=1}^{N}\lambda_i\mathcal{N}(\theta;\theta_i,\Sigma_i)d\theta \\ &= \lim_{\delta\to\infty}\sum_{i=1}^{N}\lambda_i\int p(y=c|\delta x,\theta)\mathcal{N}(\theta;\theta_i,\Sigma_i)d\theta \\ &= \lim_{\delta\to\infty}\sum_{i=1}^{N}\lambda_i p_i(y=c|\delta x,\mathcal{D}) \\ &\leq \sum_{i=1}^{N}\frac{\lambda_i}{1+\sum_{j\neq c}\exp\{-\frac{||w_i^{(j)}||}{s_{min}(J_i^{(j)})\sqrt{\frac{\pi}{8}\lambda_{min}(\Sigma_i)}}-\frac{||w_i^{(c)}||}{s_{min}(J_i^{(c)})\sqrt{\frac{\pi}{8}\lambda_{min}(\Sigma_i)}}\}}\end{aligned} \tag{25}$$

Letting

$$t_i^{(k)}=\frac{||w_i^{(k)}||}{s_{min}(J_i^{(k)})\sqrt{\frac{\pi}{8}\lambda_{min}(\Sigma_i)}}\quad k=1,2,\cdots,C$$

we have

$$\lim_{\eta\to\infty} p_{PE}(y=c|\eta x)\leq\sum_{i=1}^{N}\frac{\lambda_i}{1+\sum_{j\neq c}\exp\{-t_i^{(j)}-t_i^{(c)}\}} \tag{26}$$

□

C ADAPTIVE UNCERTAINTY-GUIDED ENSEMBLE LEARNING PROPOSITION

C.1 PROOF OF PROPOSITION 3.6

Proof. In fact, let f be any classifier with input x and denote y as the corresponding label. Following Hellman and Raviv [1970], we have

$$Pr(y\neq f(x))\leq\frac{\mathcal{H}(y)-MI(x,y)}{2}=\frac{1}{2}\mathcal{H}(y|x) \tag{27}$$

where $MI(x,y)$ is the mutual information between x and y. Note that $\mathcal{H}(y|x)=\mathcal{H}[\mathbb{E}_\theta[p(y|x,\theta)]]$ is the total uncertainty, which is positively correlated with the prediction error. It indicates that minimizing the total uncertainty can lead to a better prediction error bound. Since total uncertainty is the sum of epistemic uncertainty and irreducible aleatoric uncertainty, the epistemic uncertainty is also positively correlated with the prediction error. □

D DERIVATIONS FOR MOG REFINEMENT

In this section, we provide detailed derivations for the E-step and M-step.

E-step: construct the expected loss function of latent variable Z. Based on Eq. (12) of the main body of the paper.

$$\begin{aligned} Q(\phi|\phi^0,\mathcal{D}) &= \sum_{m=1}^{M}\sum_{i=1}^{N} p(Z=i|\mathcal{D}_m,\phi^0)\log\frac{p(\mathcal{D}_m,Z=i|\phi)}{p(Z=i|\mathcal{D}_m,\phi^0)} \\ &\propto \sum_{m=1}^{M}\sum_{i=1}^{N} p(Z=i|\mathcal{D}_m,\phi^0)\log p(\mathcal{D}_m,Z=i|\phi) \\ &= \sum_{m=1}^{M}\sum_{i=1}^{N} p(Z=i|\mathcal{D}_m,\phi^0)[\log p(\mathcal{D}_m|Z=i,\phi)+\log p(Z=i|\phi)] \\ &= \sum_{m=1}^{M}\sum_{i=1}^{N} p(Z=i|\mathcal{D}_m,\phi^0)[\log p(y_m|x_m,\theta_i,\Sigma_i)+\log\lambda_i] \end{aligned} \tag{28}$$

where

$$\begin{aligned} p(Z=i|\mathcal{D}_m,\phi^0) &= \frac{p(\mathcal{D}_m|Z=i,\phi^0)p(Z=i|\phi^0)}{\sum_j p(\mathcal{D}_m|Z=j,\phi^0)p(Z=j|\phi^0)} = \frac{\lambda_i^0 p(y_m|x_m,Z=i,\phi^0)}{\sum_j \lambda_j^0 p(y_m|x_m,Z=j,\phi^0)} \\ &= \frac{\lambda_i^0 p(y_m|x_m,\theta_i^0,\Sigma_i^0)}{\sum_j \lambda_j^0 p(y_m|x_m,\theta_j^0,\Sigma_j^0)} = \frac{\lambda_i^0\int p(y_m|x_m,\theta)N(\theta;\theta_i^0,\Sigma_i^0)d\theta}{\sum_j \lambda_j^0\int p(y_m|x_m,\theta)N(\theta;\theta_j^0,\Sigma_j^0)d\theta} \end{aligned} \tag{29}$$

and

$$p(y_m|x_m,\theta_i,\Sigma_i) = \int p(y_m|x_m,\theta)N(\theta;\theta_i,\Sigma_i)d\theta \tag{30}$$

which can be approximated either by MC sampling or probit approximation shown in Eq. (5).

M-step: obtain the parameters ϕ by maximizing $Q(\phi|\phi^0,\mathcal{D})$, which include optimizing $\{\lambda_i\}_{i=1}^N$, $\{\theta_i\}_{i=1}^N$, and $\{\Sigma_i\}_{i=1}^N$

M-step for $\{\lambda_i\}_{i=1}^N$ Conditioned on $\sum_{i=1}^N \lambda_i = 1$, we add a Lagrangian multiplier with coefficient α to $Q(\phi|\phi^0,\mathcal{D})$ to solve the constrained problem.

$$\hat{Q}(\phi|\phi^0,\mathcal{D}) = \sum_{m=1}^{M}\sum_{i=1}^{N} p(Z=i|\mathcal{D}_m,\phi^0)[\log p(y_m|x_m,\theta_i,\Sigma_i)+\log\lambda_i] - \alpha(\sum_{i=1}^{N}\lambda_i - 1) \tag{31}$$

To learn $\{\lambda_i\}_{i=1}^N$, we force the gradients of $\hat{Q}(\phi|\phi^0,\mathcal{D})$ with respect to $\{\lambda_i\}_{i=1}^N$ and α equal to 0 shown in Eq. (32).

$$\begin{aligned} \frac{\partial\hat{Q}}{\partial\lambda_i} &= \sum_{m=1}^{M}\frac{p(Z=i|\mathcal{D}_m,\phi^0)}{\lambda_i} - \alpha = 0 \qquad i=1,2,...,N \\ \frac{\partial\hat{Q}}{\partial\alpha} &= \sum_{i=1}^{N}\lambda_i - 1 = 0 \end{aligned} \tag{32}$$

Eq. (32) indicates

$$\lambda_i^* = \frac{\sum_{m=1}^{M} p(Z=i|\mathcal{D}_m,\phi^0)}{\sum_{m=1}^{M}\sum_{j=1}^{N} p(Z=j|\mathcal{D}_m,\phi^0)} \tag{33}$$

M-step for $\{\theta_i\}_{i=1}^N$ Based on Eq. (28), we can observe that maximizing $Q(\phi|\phi^0,\mathcal{D})$ with respect to θ is equal to maximizing the $Q(\theta_i|\phi^0,\mathcal{D})$ independently. $Q(\theta_i|\phi^0,\mathcal{D})$ is shown in Eq. (34).

$$Q(\theta_i|\phi^0,\mathcal{D}) = \sum_{m=1}^{M} p(Z=i|\mathcal{D}_m,\phi^0)\log p(y_m|x_m,\theta_i,\Sigma_i) \tag{34}$$

where $p(Z=i|\mathcal{D}_m,\phi^0)$ is the membership weight of data pair (x_m,y_m) belonging to the ith ensemble component $N(\theta_i,\Sigma_i)$. Noting that Σ_i can always be computed by Laplacian approximation in a post-processing manner in our framework, we only need to optimize θ_i in a deterministic way and the loss function is shown in Eq. (35).

$$\hat{Q}(\theta_i|\phi^0,\mathcal{D})=\sum_{m=1}^{M}p(Z=i|\mathcal{D}_m,\phi^0)\log p(y_m|x_m,\theta_i) \tag{35}$$

where $p(y_m|x_m,\theta_i)$ is the softmax probability generated directly by ith ensemble component. Due to the uncertainty-guided ensemble training strategy, different ensemble models will focus on different samples, leading to different $p(Z=i|\mathcal{D}_m,\phi^0), i=1,2,...,N$. Directly optimizing $\hat{Q}(\theta_i|\phi^0,\mathcal{D})$ will strengthen the samples that each model focuses on, which implicitly enhances the diversity. To further improve the diversity, we can assign each data sample to its top l nearest component based on $p(Z=i|\mathcal{D}_m,\phi^0)$. For example, let's assume $p(Z=1|\mathcal{D}_m,\phi^0)>p(Z=2|\mathcal{D}_m,\phi^0)>\cdots>p(Z=N|\mathcal{D}_m,\phi^0)$ and $l=2$. We will assign (x_m,y_m) to the first and second ensemble components. Then we can fine-tune each ensemble model with a higher concentration of the data samples they receive by performing the stochastic gradient ascent. The loss function is shown in Eq. (36)

$$\theta_i^*=\arg\max_{\theta_i}\sum_{m=1}^{M}softmax(I_l[p(Z=i|\mathcal{D}_m,\phi^0)])\log p(y_m|x_m,\theta_i) \tag{36}$$

where I_l is the indicator function, which returns 1 if $p(Z=i|\mathcal{D}_m,\phi^0)$ is the top l largest among all $\{p(Z=j|\mathcal{D}_m,\phi^0)\}_{j=1}^N$ and returns 0 otherwise. The softmax function is applied for each batch of the data to ensure that the sum of the weights equals to 1, which is similar to Eq. (11) of the main body of the paper.

M-step for $\{\Sigma_i\}_{i=1}^N$ Once we have θ_i^*, we perform the LA to get Σ_i^*.

E EXPERIMENT SETTINGS AND IMPLEMENTATION

E.1 MODEL ARCHITECTURE AND HYPERPARAMETERS

For the MNIST dataset, we use the same architecture as Gal et al. [2017]: Conv2D-Relu-Conv2D-Relu-MaxPool2D-Dropout-Dense-Relu-Dropout-Dense-Softmax. Each convolutional layer contains 32 convolution filters with 4×4 kernel size. We use a max-pooling layer with a 2×2 kernel, a dense layer with 128 units, and the dropout probability 0.5. For the CIFAR-10 dataset, we use ResNet18. During training the MNIST and CIFAR-10 models, we randomly split 10% of the training data as validation data for model selection and save the best model for uncertainty quantification. We use the SGD optimizer with an initial learning rate of 0.1 and momentum of 0.9 for both MNIST and CIFAR-10. For CIFAR-10, we decrease the learning rate to 0.01,0.001 at the 30th and 60th epochs while there is no learning rate decrease for MNIST. For MNIST, the batch size is set to 128 and the maximum epoch is 50. For CIFAR-10, the batch size is 128 and the maximum epoch is 100. We perform the standard data augmentation techniques for CIFAR-10 dataset including random cropping and random horizontal flipping. For constructing the probabilistic ensemble, we use the last-layer LA implemented by Daxberger et al. [2021], which can be found at `https://github.com/AlexImmer/Laplace`. We generate 200 samples from the mixture of Gaussian model for uncertainty quantification. Regarding to the uncertainty-guided ensemble learning strategy, we use $a=0.1, b=1$ as hyperparameters. For the MoG refinement, we choose $l=2$ for the PE model with 5 components. All the ensemble models have size 5. Each experiment is conducted over 3 independent runs and the standard derivations are also reported. We utilize an RTX2080Ti GPU to do the experiments and the proposed method is implemented using Pytorch.

E.2 IMPLEMENTATION DETAILS

In this section, we will discuss the implementation details for different uncertainty estimation methods. We use the default hyperparameters in their open-source codes except the hyperparameters mentioned in Appendix E.1.

- Ensemble: we train ensemble models with random initialization following the experiment settings in Appendix E.1.
- Batch Ensemble: the open-source code can be found in `https://github.com/giannifranchi/LP_BNN`.
- Hyper Ensemble: we train the ensemble models by varying both the initialization and the weight decay coefficients following the implementation in `https://github.com/google/uncertainty-baselines/blob/main/baselines/notebooks/Hyperparameter_Ensembles.ipynb`.

- Bayesian Ensemble: we follow the open-source code in https://github.com/TeaPearce/Bayesian_NN_Ensembles.
- LPBNN: the open-source code can be found in https://github.com/giannifranchi/LP_BNN.
- LA: we use the last-layer LA with full Hessian matrix computation as discussed in Appendix A. We use the existing software proposed by Daxberger et al. [2021], which can be found at https://github.com/AlexImmer/Laplace
- Multi-SWAG: we utilize the implementation provided by https://github.com/izmailovpavel/understandingbdl.

The model architecture, training strategy, and data transformation are the same for all baselines. The specific hyperparameters for ensemble baselines are chosen following their open-source codes. For the implementation of the probabilistic ensemble, the code will be made public after the acceptance of the paper.

Table 1: Within-dataset performance for ACC(%), NLL($\times 10^{-1}$), ECE($\times 10^{-2}$), BS ($\times 10^{-3}$) on MNIST and CIFAR-10. Each experiment result is aggregated over 3 independent runs.

Method	MNIST				CIFAR-10			
	ACC	NLL	ECE	BS	ACC	NLL	ECE	BS
Ours	99.34 $\pm$0.01	0.363 $\pm$0.00	0.89 $\pm$0.03	1.4 $\pm$0.00	**95.28** $\pm$0.05	**1.44** $\pm$0.00	**0.35** $\pm$0.03	**7.0** $\pm$0.01
Ensemble	99.29 $\pm$0.00	0.319 $\pm$0.02	1.39 $\pm$0.29	1.3 $\pm$0.05	94.63 $\pm$0.24	1.70 $\pm$0.12	0.75 $\pm$0.23	7.9 $\pm$0.43
Batch Ensemble	98.92 $\pm$0.04	0.352 $\pm$0.04	**0.25** $\pm$0.05	1.7 $\pm$0.11	92.66 $\pm$0.14	2.49 $\pm$0.03	3.04 $\pm$0.11	11.3 $\pm$0.09
Hyper Ensemble	99.32 $\pm$0.03	0.293 $\pm$0.00	1.13 $\pm$0.05	1.3 $\pm$0.04	95.13 $\pm$0.09	1.49 $\pm$0.04	0.53 $\pm$0.18	7.2 $\pm$0.17
Bayesian Ensemble	99.13 $\pm$0.07	0.348 $\pm$0.01	1.22 $\pm$0.06	1.5 $\pm$0.03	93.94 $\pm$1.08	1.90 $\pm$0.42	0.93 $\pm$0.29	8.9 $\pm$1.70
LPBNN	98.93 $\pm$0.04	0.350 $\pm$0.02	0.27 $\pm$0.08	1.7 $\pm$0.07	93.43 $\pm$0.40	2.30 $\pm$0.14	2.95 $\pm$0.45	10.3 $\pm$0.58
LA	98.51 $\pm$0.04	0.512 $\pm$0.02	0.62 $\pm$0.28	2.3 $\pm$0.10	93.31 $\pm$0.08	2.21 $\pm$0.02	2.03 $\pm$0.11	10.3 $\pm$0.11
Multi-SWAG	**99.37** $\pm$0.02	**0.228** $\pm$0.00	0.51 $\pm$0.01	**1.1** $\pm$0.02	93.71 $\pm$0.41	1.76 $\pm$0.19	0.54 $\pm$0.08	8.8 $\pm$0.53

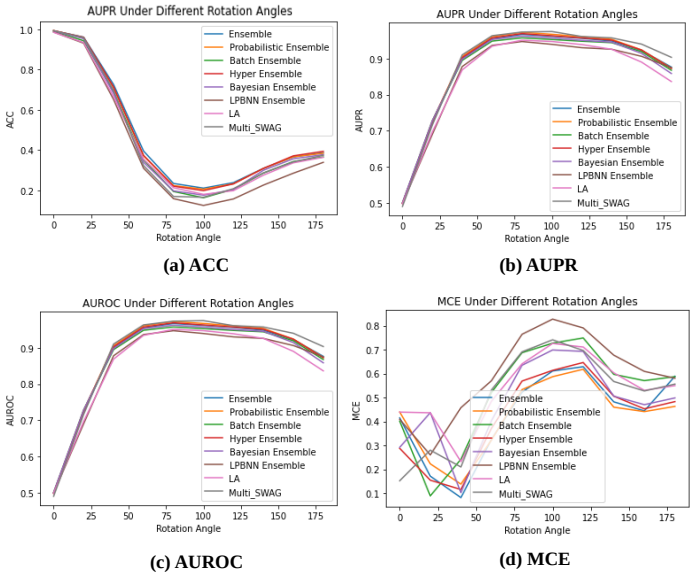
F UNCERTAINTY CALIBRATION UNDER DISTRIBUTIONAL SHIFTS

F.1 WITHIN-DATASET PERFORMANCE

The within-dataset performance for different uncertainty quantification methods can be found in Table 1. From Table 1, we can observe that our proposed method has the best within-dataset performance on CIFAR-10 for various metrics. Specifically, we achieve 15% NLL improvement and 53% ECE improvement on CIFAR-10. For MNIST, our method has comparable results with other ensemble-based methods. When there is a distributional shift between the training data and testing data for more complex datasets, our proposed method is more likely to gain significant performance enhancement. In contrast, for simpler datasets with small within-dataset shifts, our proposed method is comparable to others.

F.2 ADDITIONAL RESULTS ON DIFFERENT CALIBRATION METRICS

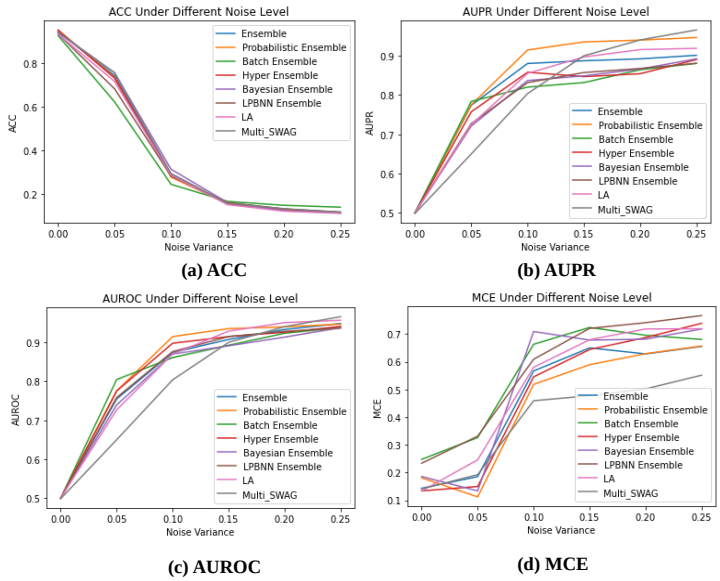
In this section, we will show additional results with uncertainty calibration metrics AUROC, AUPR, MCE, and ACC for both rotated MNIST and corrupted CIFAR-10 datasets. We can also observe from Figure 1, 2 that our proposed method outperforms others for different uncertainty calibration metrics, especially for CIFAR-10 dataset.



(a) ACC **(b) AUPR**

(c) AUROC **(d) MCE**

Figure 1: Additional uncertainty calibration metrics for rotated MINST dataset.



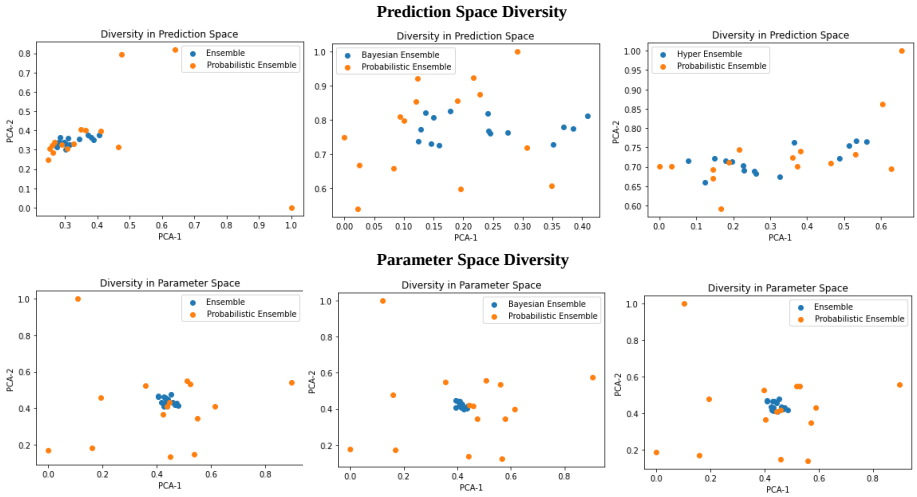
(a) ACC **(b) AUPR**

(c) AUROC **(d) MCE**

Figure 2: Additional uncertainty calibration metrics for corrupted CIFAR-10 dataset.

G VISUALIZATIONS OF DIVERSITY ANALYSIS

In this section, we also provide some visualizations for both parameter space diversity and prediction space diversity. Basically, we plot the neural network parameters as well as the predictive logits for MNIST testing data into a two-dimensional space using PCA. The baseline methods for comparison include Ensemble (ESB), Hyper Ensemble (Hyper-E), and Bayesian Ensemble (Bayesian-E). The visualizations are shown in Figure 3 pairwisely, which empirically demonstrate that our proposed method can achieve better diversity by the probabilistic ensemble with uncertainty-guided ensemble learning. Although the Bayesian ensemble and hyperparameter ensemble can also enhance the diversity, we can still observe a significant improvement using our proposed method.



(a) PE vs ESB **(b) PE vs Bayes-E** **(c) PE vs Hyper-E**

Figure 3: The visualizations of the prediction space (the first row) and parameter space (the second row) diversity. The comparison is conducted pairwisely between our proposed method with other ensemble-based methods.

H ABLATION STUDIES

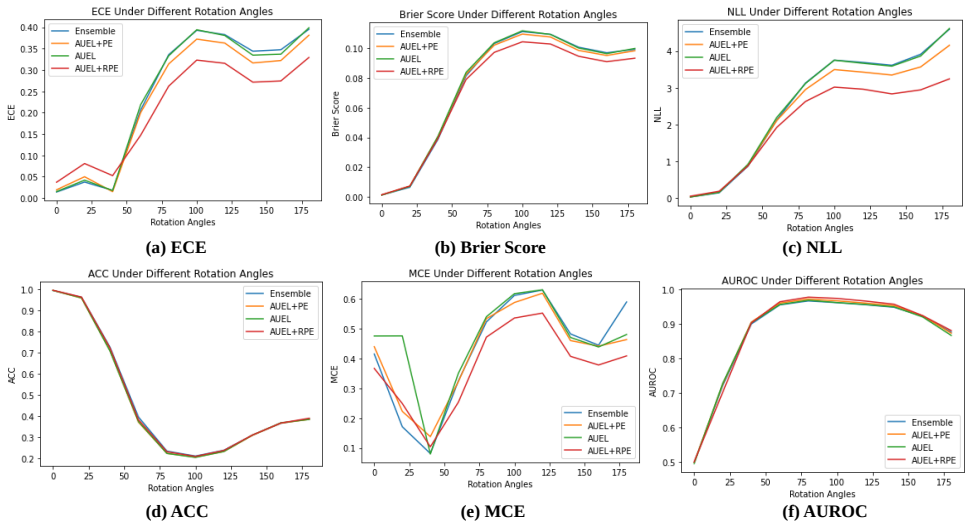
H.1 EFFECTIVENESS OF SUB-MODULES

In this section, we provide supplementary experiment results regarding the OOD detection and uncertainty calibration to further demonstrate the effectiveness of each sub-module, which are shown in Table 2, Figure 4, and Figure 5. Typically, the AUEL module can more significantly improve the performances on CIFAR-10 dataset than on MNIST dataset. This is mainly because MNIST dataset is simpler such that the training samples can all achieve small uncertainties with little differences among them. The PE module could improve the performances on both datasets with various metrics, especially for AUROC, AUPR, ECE and NLL. The refinement of MoG parameters works better on MNIST dataset and shows marginal improvements on CIFAR-10 dataset. This is reasonable since the MoG refinement can further improve the expertise of each ensemble component for MNIST dataset. However, for CIFAR-10, the AUEL already enhances the specialty of each ensemble component and it is not necessary to perform the refinement. Hence, the MoG refinement performs better on simpler datasets where the within-dataset uncertainties are all small and similar.

Table 2: Effectiveness of sub-modules: additional OOD detection results for AUROC (%) and AUPR (%) on MNIST-related and C10-related datasets with epistemic uncertainty (EU)

Method	MNIST → EMNIST		MNIST → KMNIST	
	AUROC	AUPR	AUROC	AUPR
ESB	96.15 ± 0.17	94.81 ± 0.19	96.59 ± 0.17	95.31 ± 0.20
AUEL	96.14 ± 0.20	94.74 ± 0.59	96.76 ± 0.39	95.64 ± 0.78
AUEL + PE	96.49 ± 0.04	95.36 ± 0.07	97.02 ± 0.12	96.06 ± 0.19
AUEL+RPE	**96.81** ± 0.09	**96.09** ± 0.19	**97.46** ± 0.17	**96.64** ± 0.21

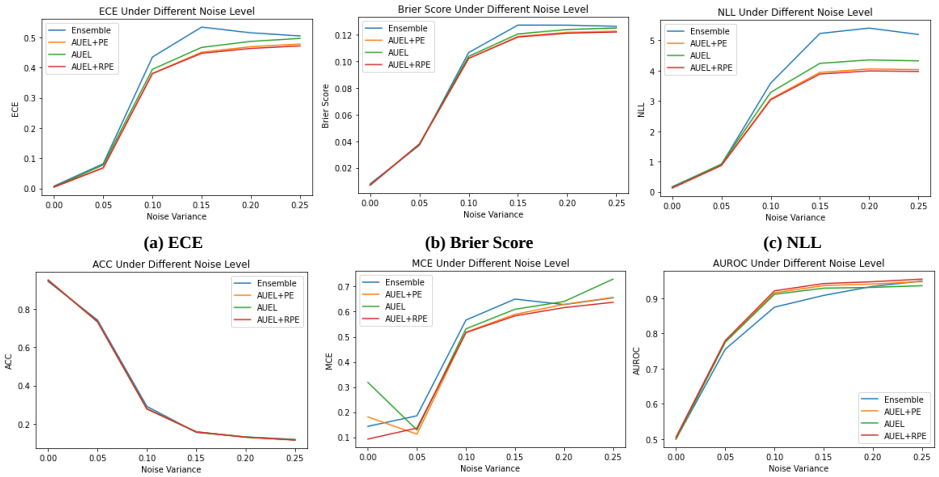
Method	C10 → LSUN		C10 → C100	
	AUROC	AUPR	AUROC	AUPR
ESB	88.42 ± 0.85	84.99 ± 0.65	91.87 ± 0.58	88.69 ± 0.55
AUEL	89.16 ± 0.12	85.55 ± 0.18	92.73 ± 0.16	89.71 ± 0.57
AUEL + PE	89.57 ± 0.08	86.81 ± 0.14	93.80 ± 0.11	91.67 ± 0.36
AUEL+RPE	**89.58** ± 0.11	**86.86** ± 0.18	**93.93** ± 0.13	**91.93** ± 0.39



(a) ECE **(b) Brier Score** **(c) NLL**

(d) ACC **(e) MCE** **(f) AUROC**

Figure 4: Effectiveness of sub-modules: additional uncertainty calibration results for rotated MNIST dataset with various metrics such as ECE, Brier Score, NLL, ACC, MCE, and AUROC.



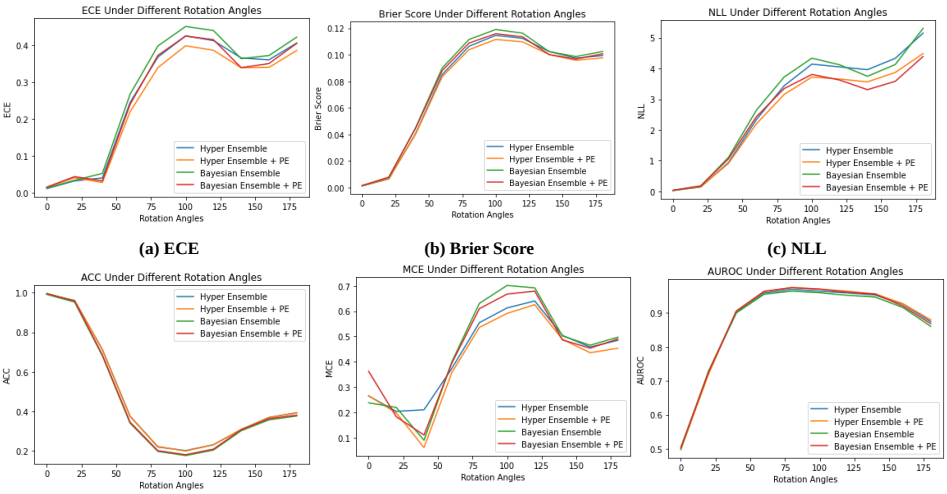
(a) ECE **(b) Brier Score** **(c) NLL**

(d) ACC **(e) MCE** **(f) AUROC**

Figure 5: Effectiveness of sub-modules: additional uncertainty calibration results for noisy CIFAR-10 dataset with various metrics such as ECE, Brier Score, NLL, ACC, MCE, and AUROC.

H.2 PROBABILISTIC ENSEMBLE AS A PLUG-AND-PLAY MODULE

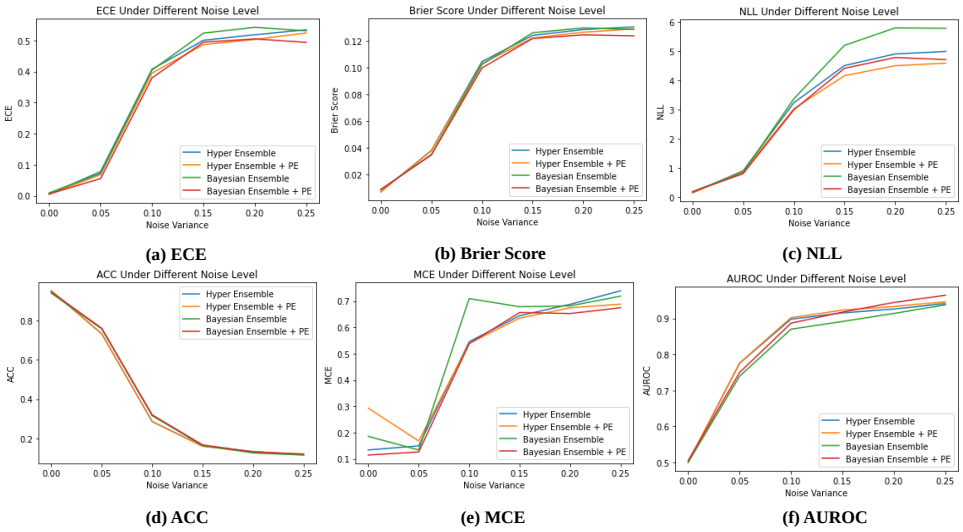
In this section, we treat the probabilistic ensemble as a plug-and-play module and add it to Bayes-E and Hyper-E to show further improvements for both OOD detection and uncertainty calibration performances. Additional experiment results are shown in Table 3 for OOD detection and Figure 6, 7 for uncertainty calibration under distributional shifts.



(a) ECE **(b) Brier Score** **(c) NLL**

(d) ACC **(e) MCE** **(f) AUROC**

Figure 6: Probabilistic ensemble as a plug-and-play module: additional uncertainty calibration results for rotated MNIST dataset with various metrics such as ECE, Brier Score, NLL, ACC, MCE, and AUROC.



(a) ECE **(b) Brier Score** **(c) NLL**

(d) ACC **(e) MCE** **(f) AUROC**

Figure 7: Probabilistic ensemble as a plug-and-play module: additional uncertainty calibration results for noisy CIFAR-10 dataset with various metrics such as ECE, Brier Score, NLL, ACC, MCE, and AUROC.

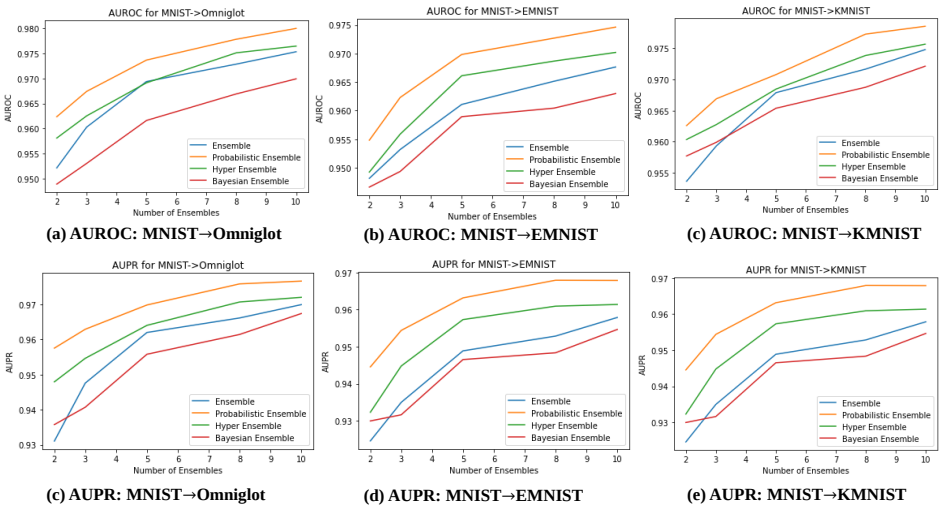
Table 3: Probabilistic ensemble as a plug-and-play module: additional OOD detection results for AUROC (%) and AUPR (%) on MNIST-related and C10-related datasets with epistemic uncertainty (EU).

Method	MNIST → EMNIST		MNIST → KMNIST	
	AUROC	AUPR	AUROC	AUPR
Bayes-E	95.55 ± 0.21	94.22 ± 0.19	96.49 ± 0.07	95.18 ± 0.09
Bayes-E + PE	**96.10** ± 0.21	**95.15** ± 0.08	**96.74** ± 0.13	**95.74** ± 0.07
Hyper-E	96.49 ± 0.04	95.36 ± 0.07	97.02 ± 0.12	96.06 ± 0.19
Hyper-E + PE	**96.76** ± 0.23	**96.03** ± 0.35	**97.24** ± 0.25	**96.64** ± 0.31

Method	C10 → LSUN		C10 → C100	
	AUROC	AUPR	AUROC	AUPR
Bayes-E	87.85 ± 1.22	84.56 ± 1.01	91.80 ± 0.45	88.83 ± 0.02
Bayes-E + PE	**89.17** ± 0.22	**87.13** ± 0.40	**94.07** ± 0.64	**92.50** ± 1.42
Hyper-E	88.82 ± 0.15	85.29 ± 0.25	92.59 ± 0.24	89.65 ± 0.71
Hyper-E + PE	**89.25** ± 0.14	**86.46** ± 0.42	**93.63** ± 0.37	**91.52** ± 0.86

H.3 EFFICIENCY ANALYSIS

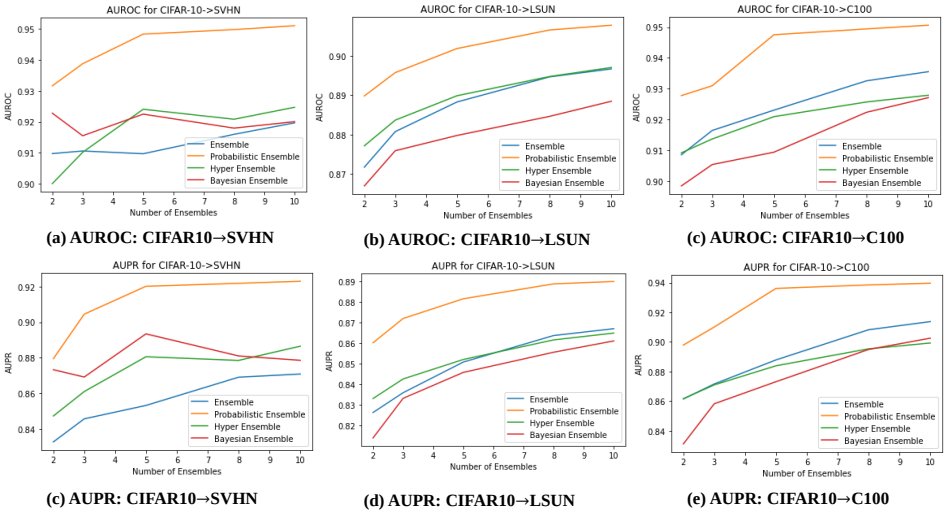
In this section, we first show the empirical runtime of our proposed method. Since AUEL requires sequential training, it is not straightly comparable to the parallel ensemble methods. However, in terms of training one ensemble component, our method performs similarly compared to the deep ensemble method. It takes 2s/35s for training one epoch of MNIST/C10 dataset. Empirically, constructing the last-layer LA for a single network takes 3.2s for MNIST and 14.4s for C10. Note that we can generate an arbitrary number of samples from the mixture of Gaussian for uncertainty quantification. If we generate 200 samples, the uncertainty estimation runtime for 10000 testing images of MNIST/C10 is 3.8s/15.4s for our method. It takes about 0.03s to obtain one more sample for uncertainty quantification. It is efficient since we only sample the last layer parameters and reuse the intermediate outputs. Moreover, we also provide the OOD detection results of the ensemble-based methods on different ensemble sizes for MNIST-related and CIFAR-related datasets. The baseline methods include ESB, Hyper-E, and Bayes-E. The results are shown in Figure 8 and Figure 9. With limited computational resources, we only need to construct the probabilistic ensemble model with a small size to achieve competitively compared to other ensemble-based methods with large sizes. Sometimes, PE model with 2 components can even achieve better performances compared to other ensemble-based methods with 10 components, i.e., C10 → SVHN shown in Figure 8 (a).



(a) AUROC: MNIST→Omniglot **(b) AUROC: MNIST→EMNIST** **(c) AUROC: MNIST→KMNIST**

(c) AUPR: MNIST→Omniglot **(d) AUPR: MNIST→EMNIST** **(e) AUPR: MNIST→KMNIST**

Figure 8: Efficiency of Probabilistic Ensemble: OOD detection results for MNIST-related datasets with metrics AUROC and AUPR.



(a) AUROC: CIFAR10→SVHN **(b) AUROC: CIFAR10→LSUN** **(c) AUROC: CIFAR10→C100**

(c) AUPR: CIFAR10→SVHN **(d) AUPR: CIFAR10→LSUN** **(e) AUPR: CIFAR10→C100**

Figure 9: Efficiency of Probabilistic Ensemble: OOD detection results for CIFAR10-related datasets with metrics AUROC and AUPR.

I APPLICATION TO LARGER DATASETS

The proposed method can be scaled up to larger datasets and larger models. We have the same complexity as the deep ensemble method during training. In the probabilistic ensemble framework, the only additional cost is LA. We use the last-layer LA that constructs a posterior approximation of the last-layer parameters, which is cost-efficient (15s for a Resnet18 model on CIFAR-10). The inference complexity of last-layer LA with a full covariance matrix is $O(m + c^3 + p^3)$ where m, c, p represent the total number of parameters, the number of classes, and the number of last-layer parameters, respectively. The memory complexity is $O(m + c^2 + p^2)$. In this section, we apply our proposed method to CIFAR-100 (C100) and TinyImagenet (TIM) with empirical results.

We utilize Resnet18 as the backbone for both datasets. The training hyperparameters are illustrated below. For CIFAR-100,

the maximum epoch is set to be 200 and the batch size is 128. We use an SGD optimizer with an initial learning rate of 0.1 and momentum of 0.9. During training, the learning rate decreases to 0.01, 0.001, 0.0001 at the 50th, 100th, and 150th epoch. For TinyImagenet, the maximum epoch is 80 with batch size 128. We use the same optimizer as CIFAR-100 with learning rate decay at the 20th, 40th, and 60th epoch. The standard data augmentation is conducted for both datasets including random cropping and random horizontal flipping. We randomly select 10% of the training data as validation data for CIFAR-100 while the validation data is of TinyImagenet is provided. To construct the probabilistic ensemble model after training, we follow the same experiment settings shown in Appendix Sec. E.1.

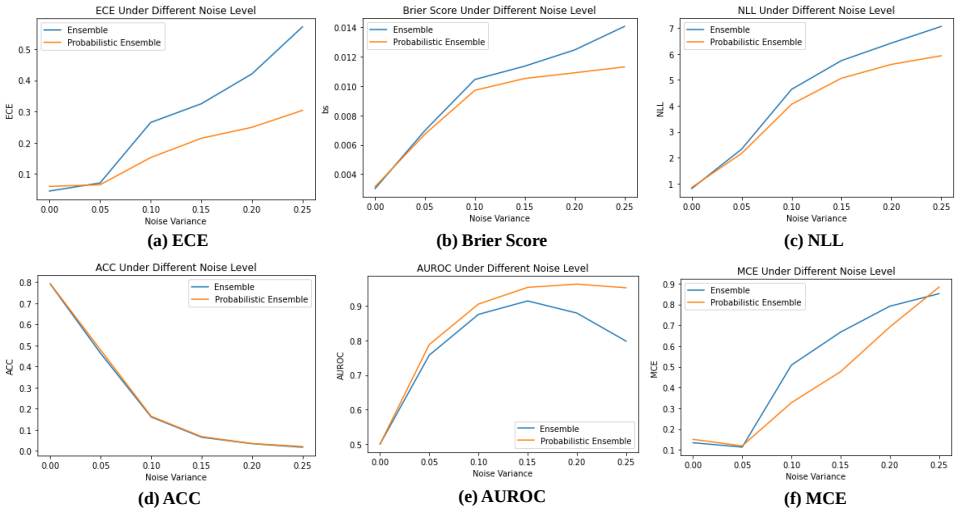
During the evaluation, we show the OOD detection results and the uncertainty calibration performance under distributional shifts, respectively. For CIFAR-100 and TinyImagenet, we use LSUN, SVHN, and CIFAR-10 as the OOD datasets. We use the same evaluation metrics illustrated in Sections 4.1 and 4.2 of the main body of the paper. We compare the proposed AUEL+PE with the deep ensemble method since the deep ensemble method is the most representative ensemble-based method with competitive performance. The experiments are conducted on CIFAR-100 and TinyImagenet testing datasets for OOD detection. The uncertainty calibration evaluation is conducted on the validation dataset of Tinyimagenet since TIM testing data does not provide labels. To create the corrupted C100 and corrupted TIM datasets, we add the Gaussian noise with 0 mean and variance ranging from 0 to 0.25 with a step of 0.05 to the original datasets following Sec. 4.2 of the main body of the paper.

The experiment results are shown in Table 4 for OOD detection and Figures 10, 11 for uncertainty calibration performance. With the enhanced diversity of our proposed method, we can consistently outperform the deep ensemble method for both tasks. For OOD detection, we also observe that AUEL+PE is more stable with small standard derivations. For the uncertainty calibration under distributional shifts, we basically obtain increasing performance improvement as the shifts become more significant, indicating the better generalization ability enforced by improved diversity.

Table 4: OOD Detection Results for AUROC (%) and AUPR (%) on CIFAR-100 and TinyImagenet with Epistemic Uncertainty. Each experiment result is aggregated over 3 independent runs. The standard derivation is also reported.

Method	C100 → SVHN		C100 → LSUN		C100→ C10	
	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
AUEL+PE	**83.56** ± 1.61	**73.97** ± 2.61	**79.02** ± 0.11	**73.28** ± 0.20	**87.35** ± 0.34	**84.20** ± 0.45
Ensemble	77.53 ± 4.25	71.06 ± 4.56	74.97 ± 2.85	69.68 ± 2.24	79.37 ± 2.77	74.16 ± 2.08

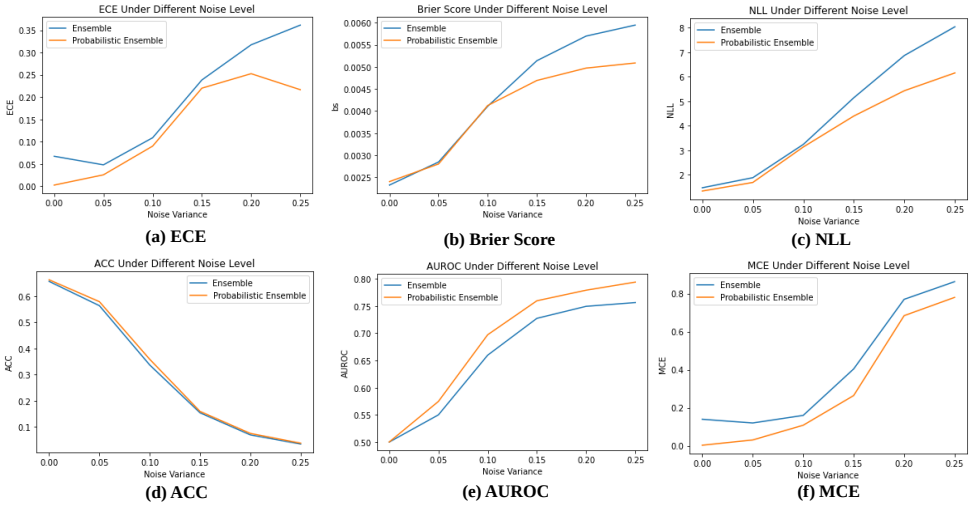
Method	TIM → SVHN		TIM → LSUN		TIM→ C10	
	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
AUEL+PE	**80.84** ± 2.66	**72.15** ± 3.73	**86.71** ± 0.43	**82.67** ± 0.83	**87.07** ± 0.47	**83.60** ± 0.45
Ensemble	78.52 ± 1.58	71.08 ± 4.67	79.21 ± 1.14	78.01 ± 2.25	84.92 ± 1.19	81.06 ± 0.96



(a) ECE (b) Brier Score (c) NLL

(d) ACC (e) AUROC (f) MCE

Figure 10: Uncertainty calibration results for corrupted CIFAR-100 dataset with various metrics such as ECE, Brier Score, NLL, ACC, MCE, and AUROC.



(a) ECE **(b) Brier Score** **(c) NLL**

(d) ACC **(e) AUROC** **(f) MCE**

Figure 11: Uncertainty calibration results for corrupted TinyImagenet dataset with various metrics such as ECE, Brier Score, NLL, ACC, MCE, and AUROC.

J OTHER DISTRIBUTIONAL SHIFTS

Besides generating shifted distribution by adding Gaussian noise, we further perform different adversarial attacks and evaluate the robustness of our proposed methods. We generate adversarial samples x_{adv} for each C10 testing image x following the fast gradient sign method (FGSM) shown in Eq. (37).

$$x_{adv} = x + \epsilon \text{sign}(\nabla_x L(\theta, x, y)) \tag{37}$$

where L is the NLL loss, ϵ is a hyperparameter indicating the perturbation level, sign(u) is function that outputs 1 if $u >= 0$ and outputs -1 if $u < 0$. Then we compute the ACC and NLL of our proposed methods on the perturbed images, compared to various ensemble baselines. Recently, there is a newly proposed type of attacks called uncertainty attacks, which try to optimize the uncertainty of the prediction. We replace the L in Eq. (37) by the entropy of $p(y|x, \theta)$ to perform uncertainty attacks. The results are shown in Table 5, where "single" refers to single deterministic network. It indicates the effectiveness of our proposed methods on different adversarial attacks. Compared to the deep ensemble method, ours can achieve significant improvement, especially when ϵ is small.

Table 5: The ACC and NLL under different adversarial attacks on C10 dataset.

Method	Adversarial Attacks with FGSM using NLL Loss							
	$\epsilon = 0.01$		$\epsilon = 0.02$		$\epsilon = 0.05$		$\epsilon = 0.08$	
	ACC	NLL	ACC	NLL	ACC	NLL	ACC	NLL
Ours	0.732 ± 0.01	**0.93** ± 0.02	**0.547** ± 0.02	**1.81** ± 0.17	**0.309** ± 0.01	**3.04** ± 0.23	**0.206** ± 0.01	**3.55** ± 0.12
Single	0.534 ± 0.13	4.01 ± 1.99	0.367 ± 0.10	6.57 ± 2.30	0.206 ± 0.10	7.94 ± 2.30	0.152 ± 0.05	7.66 ± 1.32
Ensemble	0.684 ± 0.07	1.31 ± 0.52	0.500 ± 0.07	2.56 ± 0.99	0.285 ± 0.04	3.98 ± 1.16	0.190 ± 0.02	4.51 ± 1.07
Hyper-E	**0.733** ± 0.01	0.94 ± 0.08	0.546 ± 0.02	1.89 ± 0.20	**0.309** ± 0.01	3.21 ± 0.30	0.200 ± 0.00	3.78 ± 0.32
Bayes-E	0.692 ± 0.03	1.14 ± 0.18	0.507 ± 0.03	2.15 ± 0.35	0.298 ± 0.02	3.46 ± 0.57	0.200 ± 0.01	3.96 ± 0.60

Method	Adversarial Attacks with FGSM using Uncertainty Loss							
	$\epsilon = 0.01$		$\epsilon = 0.02$		$\epsilon = 0.05$		$\epsilon = 0.08$	
	ACC	NLL	ACC	NLL	ACC	NLL	ACC	NLL
Ours	**0.784** ± 0.00	**0.65** ± 0.01	0.582 ± 0.00	**1.52** ± 0.02	**0.328** ± 0.00	**2.88** ± 0.01	**0.212** ± 0.00	**3.40** ± 0.02
Single	0.569 ± 0.14	3.10 ± 1.72	0.391 ± 0.10	5.61 ± 2.08	0.223 ± 0.05	7.34 ± 1.14	0.160 ± 0.02	7.40 ± 0.48
Ensemble	0.733 ± 0.07	0.91 ± 0.35	0.535 ± 0.07	2.10 ± 0.79	0.298 ± 0.04	3.72 ± 1.03	0.192 ± 0.02	4.41 ± 1.02
Hyper-E	0.783 ± 0.01	0.66 ± 0.04	**0.584** ± 0.02	1.57 ± 0.15	0.323 ± 0.01	3.03 ± 0.26	0.202 ± 0.00	3.71 ± 0.31
Bayes-E	0.738 ± 0.03	0.82 ± 0.11	0.539 ± 0.03	1.78 ± 0.25	0.309 ± 0.02	3.24 ± 0.40	0.203 ± 0.01	3.87 ± 0.58

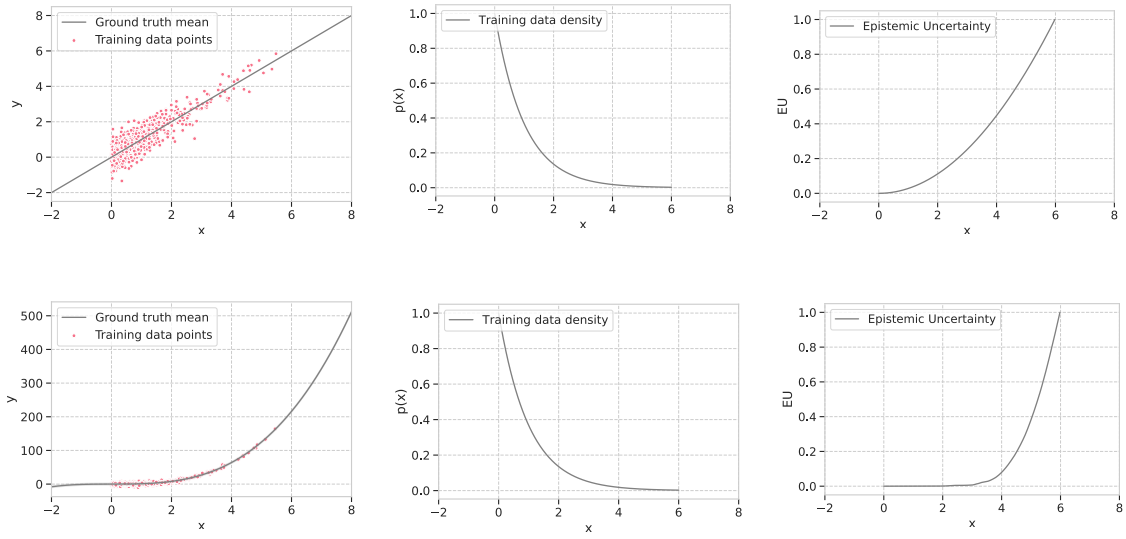


Figure 12: Synthetic data experiment. The first and second row represent the experiments for linear and nonlinear datasets, respectively. In the first column, the red dots are the training samples and the gray line is the ground truth mean of $p(y|x)$. In the second column, the ground truth input data density $p(x)$ is plotted as a function of x. The third column shows the normalized estimated epistemic uncertainty by PE as a function of input x.

K SYNTHETIC EXPERIMENTS

One-dimensional Regression Problems. To demonstrate the reliability of PE to quantify epistemic uncertainty, we provide some toy examples for one-dimensional regression problems $y = f(x) + \epsilon$ where ϵ is the noise term. We use both linear and nonlinear synthetic datasets. We first sample x in a region of $[0, 6]$ from an exponential distribution $p(x) = e^{-x}$.

Then for each x_i, the corresponding y_i is sampled from $\mathcal{N}(x_i, 0.5)$ for the linear dataset and $\mathcal{N}(x_i^3, 0.5)$ for the nonlinear dataset. We obtain 600 pairs of $\{x_i, y_i\}$ as training data for both datasets. Given the training data, we build a ReLU network using two fully-connected layers and each layer has 20 hidden nodes. The neural network outputs $p(y|x, \theta) = \mathcal{N}(\mu(x, \theta), \sigma^2(x, \theta))$ where θ are the model parameters. In the probabilistic ensemble framework, we construct a 2-component PE model where the posterior distribution of parameters $p(\theta|\mathcal{D})$ is approximated by $\sum_{i=1}^{2} 0.5\mathcal{N}(\theta; \theta_i, \Sigma_i)$ through LA with equal weights. The epistemic uncertainty of x can be calculated by

$$Var_{p(\theta|\mathcal{D})}[E_{p(y|x,\theta)}[y]] \approx \frac{1}{M-1}\sum_{j=1}^{m}[\mu(x, \theta^{(j)}) - \frac{1}{M}\sum_{j=1}^{m}\mu(x, \theta^{(j)})]^2$$

where $\theta^{(j)} \sim \sum_{i=1}^{2} 0.5\mathcal{N}(\theta; \theta_i, \Sigma_i)$. The results are shown in Figure 12.

From Figure 13, we can clearly see that the epistemic uncertainty is inversely correlated with training data density. Neural networks will not overfit the training data in the region $[0, 6]$ and training samples receive different epistemic uncertainties based on their density.

Two-moon Dataset We also apply our method to the two-moon dataset. We generate 500 two-dimensional training data points using sklearn package with noise 0.1. For each ensemble component, we use a ReLU network using two fully-connected layers and each layer has 20 hidden nodes. The network outputs logits for this two-class classification problem. For constructing the probabilistic ensemble framework, we also construct a 2-component PE model where the posterior distribution of parameters $p(\theta|\mathcal{D})$ is approximated by $\sum_{i=1}^{2} 0.5\mathcal{N}(\theta; \theta_i, \Sigma_i)$ through LA with equal weights.

The results are shown in Figure 13, which also indicates that the estimated epistemic uncertainty inversely matches with the training data density. As shown by DUQ Van Amersfoort et al. [2020], the deep ensemble method performs poorly on the

two-moon dataset. In contrast, our method performs better than the deep ensemble method.

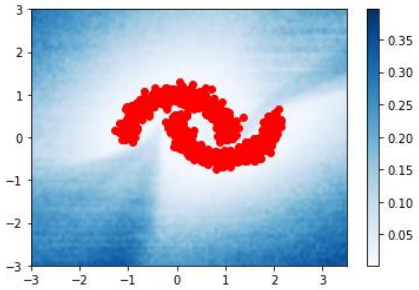


Figure 13: Synthetic data experiment on two-moon dataset. The red points are the training data. Darker regions indicate higher epistemic uncertainty.

References

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Stefan Depeweg, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning*, pages 1184–1193. PMLR, 2018.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

Andrew Gelman. Induction and deduction in bayesian data analysis. 2011.

Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019.

Martin Hellman and Josef Raviv. Probability of error, equivocation, and the chernoff bound. *IEEE Transactions on Information Theory*, 16(4):368–372, 1970.

Bas JK Kleijn and Aad W van der Vaart. The bernstein-von-mises theorem under misspecification. *Electronic Journal of Statistics*, 6:354–381, 2012.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks. In *International Conference on Machine Learning*, pages 5436–5446. PMLR, 2020.

JG Liao and Arthur Berg. Sharpening jensen's inequality. *The American Statistician*, 2018.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

Nicol N Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7): 1723–1738, 2002.

Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *International conference on machine learning*, pages 9690–9700. PMLR, 2020.