NEURAL BOOTSTRAPPER

Anonymous authors
Paper under double-blind review

ABSTRACT

Bootstrapping has been a primary tool for uncertainty quantification, and their theoretical and computational properties have been investigated in the field of statistics and machine learning. However, due to its nature of repetitive computations, the computational burden required to implement bootstrap procedures for the neural network is painfully heavy, and this fact seriously hurdles the practical use of these procedures on the uncertainty estimation of modern deep learning. To overcome the inconvenience, we propose a procedure called *Neural Bootstrapper* (NeuBoots). We reveal that the NeuBoots stably generate valid bootstrap samples that coincide with the desired target samples with minimal extra computational cost compared to traditional bootstrapping. Consequently, NeuBoots makes it feasible to construct bootstrap confidence intervals of outputs of neural networks and quantify their predictive uncertainty. We also suggest NeuBoots for deep convolutional neural networks to consider its utility in image classification tasks, including calibration, detection of out-of-distribution samples, and active learning. Empirical results demonstrate that NeuBoots is significantly beneficial for the above purposes.

1 INTRODUCTION

Since the introduction of the nonparametric bootstrap (Efron, 1979), bootstrap (or bagging) procedures have been commonly used as a primary tool in quantifying uncertainty lying on statistical inference, e.g. evaluations of standard errors, confidence intervals, and hypothetical null distribution. This success is because of its simplicity and theoretical optimality. Under moderate regularity conditions, the bootstrap procedures asymptotically approximate the sampling variability of statistical procedures, and its powerful performance in practice was confirmed in various literature (Davison and Hinkley, 1997; Efron, 2000; Hall, 1994). Despite its success in statistics field, the use of bootstrap procedures in neural network applications has been less highlighted due to its computational intensity. In uncertainty quantification, bootstrap procedures require evaluating at least hundreds of models, and this multiple training is infeasible in practice in terms of computational cost.

To utilize bootstrap for deep neural networks, we propose a novel procedure called *Neural Bootstrapper* (NeuBoots). The main idea is to construct a generator function that maps bootstrap weights to bootstrap samples. Our new procedure is motivated from a recent work, *Generative Bootstrap Sampler* (GBS) (Shin et al., 2020), in accelerating the computational speed of bootstrap procedure, but we note that our procedure is strictly different from the GBS. The GBS mainly focuses on classical models in statistics, and its application is limited to parametric models. In contrast, the NeuBoots is designed to generate a bootstrap distribution of neural net outputs, and we apply the NeuBoots to identify uncertainty in prediction via Convolutional Neural Networks (CNNs) (CireşAn et al., 2012; LeCun et al., 1998). Throughout this paper, we show that our NeuBoots enjoys multiple advantages over the existing uncertainty quantification procedures.

The NeuBoots is easily applicable to existing various neural networks with a minimal effort. By constructing a generator function whose input is bootstrap weights, neural networks with the NeuBoots procedure only require to concatenate bootstrap weights into the input vector of the target network. This means that the NeuBoots does not inject randomness into the network parameters that are usually large-numbered, but it directly generates bootstrap samples of the output of the target network. This is a clear advantage over the Bayesian approaches (Graves, 2011; Louizos and Welling, 2017). Bayesian Neural Networks (Bayesian NNs) based on variational inference focus on the posterior distribution of network parameters. However, due to the fact that the number of parameters even in a moderate-sized network is enormous, evaluating such a high-dimensional distribution is practically challenging in

terms of training time and memory resources. In contrast, the randomness of the NeuBoots stems from the input of bootstrap weights instead of the model parameters. So, the approximation of the distribution of model parameters, which is high-dimensional, is unnecessary. This property of the NeuBoots enables us to scalably compute the bootstrap distribution of the output of CNNs such as ResNet (He et al., 2016) and DenseNet (Huang et al., 2017). These are examined in Section 4.

We theoretically prove that the NeuBoots provides a valid bootstrap distribution of the neural network of interest. We first show that the vanilla version of the NeuBoots, which constructs the exact bootstrap distribution, then we adopt the block bootstrap (Carlstein et al., 1998) for scalable approximation by considering blocks of data observations. Theorem A.2 in the supplementary materials ensures that this modification is asymptotically equivalent to the conventional non-block bootstrap, and our empirical results also support this theoretical assertion.

We also apply the NeuBoots to various uncertainty estimation with image classification tasks. First, we apply our NeuBoots to Out-Of-Distribution (OOD) detection task. We evaluate some scorings from the bootstrap distribution, including the max of predictive mean, the standard deviation, predictive entropy, and expected entropy. Then, we train an OOD detector considering the scorings as its input. The details of the OOD procedure are given in Section 4. Our results show that the NeuBoots outperforms the state-of-the-art OOD procedures such as ODIN (Liang et al., 2018) and Mahalanobis distance-based method (Lee et al., 2018). Secondly, we evaluate the confidence estimation performance of NeuBoots on CIFAR and architectures in Section 4.2. The results show that the proposed methods can estimate the confidence of the prediction correctly while preserving the classification performance of the baseline models. Finally, we evaluate NeuBoots on an active learning task, one of the main application of uncertainty estimation, in Section 4.3. It shows there is a significant performance gap between the NeuBoots and the other sampling strategies, e.g. MCDrop (Gal and Ghahramani, 2016).

2 Related Work

Bootstrapping Neural Network Since Efron (1979) first proposed the nonparametric bootstrapping to quantify uncertainty in general settings, there has been a rich amount of literature that investigate theoretical advantages of using bootstrap procedures Hall (1986), Hall (1992), and Efron (1987) showed that bootstrap procedures are capable of achieving second-order correctness. That means that bootstrapped distribution converges to the target significantly faster than classical asymptotic approximations that only attain the first-order correctness. Franke and Neumann (2000) investigated bootstrap consistency of one-layered multi-layer perceptron (MLP) under some strong regularity conditions. Reed et al. (2014) considered using a conventional nonparametric bootstrapping to robustify classifiers under noisy labeling. However, due to the nature of repetitive computations, its practical application to large-sized data sets is not trivial. Nalisnick and Smyth (2017) proposed an approximation of bootstrapping for neural network by applying amortized variational Bayes. Despite its computational efficiency, the armortized bootstrap does not induce the exact target bootstrap distribution, and its theoretical justification is lacking.

Uncertainty Estimation There are many approaches to estimate the confidence intervals of prediction of the deep neural networks. Deep Confidence (Cortés-Ciriano and Bender, 2018) proposes a framework to compute confidence intervals for individual predictions using snapshot ensembling and conformal prediction. Also, a calibration procedure to approximate a confidence interval is proposed based on Bayesain NNs (Kuleshov et al., 2018). However, these approaches do not provide any theoretical guarantees. In contrast, our theory proves that the NeuBoots generates statistically valid bootstrap samples. Previously, approximate inference in Bayesian NNs has been proposed mainly in the literature. Gal and Ghahramani (2016) proposes MCDrop which captures model uncertainty casting dropout training in neural networks. Smith and Gal (2018) examines various measures of uncertainty for adversarial example detection. Instead of Bayesian NN, Lakshminarayanan et al. (2017) proposes a non-Bayesian approach for estimating predictive uncertainty based on ensembles and adversarial training. Compared to DeepEnsemble, NeuBoots does not require adversarial training nor learning multiple models, hence its training burden is affordable. Furthermore, NeuBoots does not suffer performance degradation caused by bootstrap sampling inefficiency. This is because we use a smoothed version of nonparametric bootstrap called Random Weight Bootstrap (RWB; (Shao and Tu, 1996)) that utilizes the entire data set unlike nonparametric bootstrap.

3 NEURAL BOOTSTRAPPER

In this section, we first present a reinterpretation of bootstrapping method as a *functional* on the class of neural networks, and then we extend it to a neural bootstrapping. Let us denote the training data set by $\mathcal{D}_{\text{train}} = \{(X_i, y_i)\}_{i=1}^n$, where each feature $X_i \in \mathcal{X} \subset \mathbb{R}^p$ and its response $y_i \in \mathbb{R}^d$. We denote a class of neural networks of interest by $\mathcal{N}$.

3.1 RANDOM WEIGHT BOOTSTRAPPING

First we list some notation regarding bootstrapping. Let $\mathbf{w} = (w_1, \ldots, w_n) \in \mathcal{W}_n \subset \mathbb{R}_+^n$ be bootstrap weights, where $\mathcal{W}_k = \{\mathbf{w} \in \mathbb{R}_+^k : \sum_{i=1}^k w_i = k\}$. Given $\mathcal{D}_{\text{train}} = \{(X_1, y_1), \ldots, (X_n, y_n)\}$, we define a functional $\Lambda : \mathcal{N} \times \mathcal{W} \to \mathbb{R}$ as follows:

$$\Lambda[f](\mathbf{w}) := \langle \mathbf{w}, L(f, \mathcal{D}_{\text{train}}) \rangle, \quad L(f, \mathcal{D}_{\text{train}}) := \{\ell(f(X_1), y_1), \ldots, \ell(f(X_n), y_n)\}.$$

where ℓ is an arbitrary loss function. For $b = 1, \ldots, B$, let us sample $\mathbf{w}^{(b)} \sim \mathbb{P}_{\mathcal{W}_n}$ where $\mathbb{P}_{\mathcal{W}_n}$ is a probability distribution on $\mathcal{W}_n$. Hence we can interpret $\Lambda[f]$ as a random variable defined on the probability space $(\mathcal{W}_n, \mathbb{P}_{\mathcal{W}_n})$ for a given $f \in \mathcal{N}$. Then a bootstrap sample of f is expressible as

$$\widehat{f}^{(b)} = \arg\min_{f \in \mathcal{N}} \Lambda[f](\mathbf{w}^{(b)}) = \arg\min_{f \in \mathcal{N}} \sum_{i=1}^n w_i^{(b)} \ell(f(X_i), y_i) \tag{3.1}$$

Under $\mathbb{P}_{\mathcal{W}_n} = \text{Multinomial}(n; 1/n, \ldots, 1/n)$, the resulting procedure is called Nonparametric Bootstrap (Efron, 1979). Also, as a smoothed version of this nonparametric bootstrap and a generalization of the Bayesian bootstrap (Rubin, 1981), Random Weight Bootstrap (RWB; Shao and Tu (1996)) is proposed with generalizing the weight distribution, and a common choice is $\mathbb{P}_{\mathcal{W}_n} = n \times \text{Dirichlet}(1, \ldots, 1)$ (Newton and Raftery, 1994). Let us note that unlike nonparmaetric bootstrap, the RWB fully utilizes the observed data points. It is well-known that the nonparametric bootstrap uses only 63% of observations for each bootstrap evaluation, because the corresponding multinomial weight results in some zero individual weight. On the other hand, the weight of the RWB is always nonzero, because it is generated from a continuous weight distribution, the Dirichlet distribution. As a result, none of observations is ignored in the bootstrap procedure, and this would be a clear advantage over nonparametric bootstrap. In this paper, we mainly focus on the RWB.

3.2 GENERATIVE EXTENSION OF BOOTSTRAPPING

To generate bootstrap samples based on equation 3.1, one has to train each $\widehat{f}^{(b)}$ for $b = 1, \ldots, B$ and store the parameters of each network. Furthermore, for a prediction, each network should evaluate $\widehat{f}^{(b)}(X_*)$ independently for a given data point X_*, so it requires B times exhaustive forward propagation to obtain bootstrap confidence intervals. These hurdles motivate us to develop a generative model version of bootstrapping which can generate bootstrap samples without multiple training nor forward propagation. Recently, Shin et al. (2020) proposes GBS which accelerates bootstrapping procedure for parametric models satisfying the above motivation. Note that Λ can receives $\widehat{f}^{(b)}$ as an input of the functional, so we can evaluate $\Lambda[\widehat{f}^{(b)}](\mathbf{w})$ for any $\mathbf{w} \in \mathcal{W}_n$. From this observation and the inspiration from GBS, we modify Λ to be a generative functional as follows.

Let M_β denotes the fully-connected network with parameter β in the final layer of f. Then we can decompose f into $f = M_\beta \circ F_\theta$ where F_θ is the feature extractor with parameter θ. We modify M_β to receive a supplementary input $\mathbf{w} \in \mathcal{W}_n$ as a seed of generative model. Let $\beta_{\mathbf{w}}$ denotes an additional parameters for $\mathbf{w}$ in the fully-connected layer. Write $\phi = (\theta, \beta \oplus \beta_{\mathbf{w}})$ where $\oplus$ denotes the concatenation operation. Then we define a mapping $G : \mathbb{R}^p \times \mathcal{W}_n \to \mathbb{R}^d$ such that $G_\phi(X, \mathbf{w}) := M_{\beta \oplus \beta_{\mathbf{w}}} \circ (F_\theta(X) \oplus \mathbf{w})$. Then we define $\Phi[G]$ on $\mathcal{W}_n$ such that $\Phi[G](\mathbf{w}) := \langle \mathbf{w}, L(G(\mathbf{w}), \mathcal{D}_{\text{train}}) \rangle$ where $L(G(\mathbf{w}), \mathcal{D}_{\text{train}}) = \{\ell(G(X_1, \mathbf{w}), y_1), \ldots, \ell(G(X_n, \mathbf{w}), y_n)\}$. Let $\mathcal{G}$ denote the class of these extended neural networks G and we call it the *generator* of Φ. Compared to $L(f, \mathcal{D}_{\text{train}})$, note that $L(G(\cdot), \mathcal{D}_{\text{train}})$ in $\Phi[G]$ receives additional input $\mathbf{w} \in \mathcal{W}_n$. Thus, learned G can generate bootstrap samples by plugging $\mathbf{w}$ into $G_\phi(X, \cdot)$ without repetitive forward-propagation, hence $\Phi[G]$ derives a generative version of bootstrapping in this point of view.

We optimize $\Phi[G]$ via the following new objective function:

$$\widehat{G} = \arg\min_{G \in \mathcal{G}} \mathbb{E}_{\mathbf{w} \sim \mathbb{P}_{\mathcal{W}_n}} \left[\Phi[G](\mathbf{w})\right], \tag{3.2}$$

Algorithm 1: Training step in NeuBoots.

Input : Dataset $\mathcal{D}$; epochs T; blocks S; index function u; learning rate ρ.

Initialize neural network parameter $\phi^{(0)}$ and set $n := |\mathcal{D}|$.

for $t \in \{0, \dots, T-1\}$ **do**

  Sample $\boldsymbol{\alpha}^{(t)} = \{\alpha_1^{(t)}, \dots, \alpha_S^{(t)}\} \overset{\text{i.i.d.}}{\sim} H_{\boldsymbol{\alpha}}$

  Replace $w_{\boldsymbol{\alpha}}^{(t)} = \{\alpha_{u(1)}^{(t)}, \dots, \alpha_{u(n)}^{(t)}\}$

  Update $\phi^{(t+1)} \leftarrow \phi^{(t)} - \frac{\rho}{n} \langle w_{\boldsymbol{\alpha}}^{(t)}, \nabla_\phi L(G_\phi(\boldsymbol{\alpha}^{(t)}), \mathcal{D}) \rangle \big|_{\phi=\phi^{(t)}}$.

Algorithm 2: Prediction step in NeuBoots.

Input : Data point $X_* \in \mathbb{R}^p$; number of bootstrap samples B.

Evaluate the feed-forward network $\widehat{G}_*(\cdot) = G_\phi(X_*, \cdot)$.

for $b \in \{1, \dots, B\}$ **do**

  Generate $\boldsymbol{\alpha}^{(b)} \overset{\text{i.i.d.}}{\sim} H_{\boldsymbol{\alpha}}$ and evaluate $\widehat{y}_*^{(b)} = \widehat{G}_*(\boldsymbol{\alpha}^{(b)})$.

Note that the solution of equation 3.2 coincides with the solution of equation 3.1 provided the uniqueness of the solution of equation 3.1. Then, for a feature of interest X_*, we can theoretically show that $\widehat{G}(X_*, \mathbf{w}) = \widehat{f}_{\mathbf{w}}(X_*)$ holds almost surely (see Theorem A.1 in the supplementary materials), where $\widehat{f}_{\mathbf{w}} = \arg\min_{f \in \mathcal{N}} \Lambda[f](\mathbf{w})$. This means that the bootstrapped sample is exactly matched to the conventional target that shares the same weight.

3.3 NeuBoots Algorithm

Despite of its exactness, $\Phi[G]$ receives a supplementary input $\mathbf{w}$ from high-dimension space $\mathcal{W}_n$, so its practical implementation and optimization via equation 3.2 would be hurdled for massive-sized data sets. To overcome this problem, we utilize a block bootstrapping procedure to reduce the dimension of the supplementary input. The proposed procedure asymptotically converges towards the same target distribution where the conventional non-block bootstrap converges to, and under some mild regularity conditions, this result is rigorously proven in the supplementary materials.

Block bootstrapping For $m \in \mathbb{N}$, we write $[m] := \{1, \dots, m\}$. Let $I_1, \dots, I_S$ denotes the index sets of exclusive S blocks. We allocate the index of training data $[n]$ to each block $I_1, \dots, I_S$ by the stratified sampling to balance among classes. Let index function $u : [n] \to [S]$ denotes such assignment i.e. $u(i) = s$ if $i \in I_s$. Then, for some weight distribution $H_{\boldsymbol{\alpha}}$ on $\mathcal{W}_S$, we impose the same value of weight on all elements in a block such as, $w_i = \alpha_{u(i)}$, where $\boldsymbol{\alpha} = \{\alpha_1, \dots, \alpha_S\} \sim H_{\boldsymbol{\alpha}}$ for $i \in [n]$, and we define $w_{\boldsymbol{\alpha}} = \{\alpha_{u(1)}, \dots, \alpha_{u(n)}\}$. Similar with the vanilla version of the GBS, setting $H_{\boldsymbol{\alpha}} = S \times \text{Dirichlet}(1, \dots, 1)$ induces a block version of the RWB, and imposing $H_{\boldsymbol{\alpha}} = \text{Multinomial}(S; 1/S, \dots, 1/S)$ results in a block nonparametric bootstrap. We also remark that the Dirichlet distribution with a uniform parameter of one can be easily approximated by independent exponential distribution. That is, $z_i / \sum_{k=1}^n z_k \sim \text{Dirichlet}(1, \dots, 1)$ for independent and identically distributed $z_i \sim \text{Exp}(1)$. Due to the fact that $\sum_{i=k}^n z_k / n \approx 1$ by the law of large number for a moderately large n, $n^{-1} \times \{z_1, \dots, z_n\}$ approximately follows the Dirichlet distribution. This property is convenient in a sense that we do not need to consider the dependence structure in $\mathbf{w}$, and simply generate independent samples from $\text{Exp}(1)$ to sample the bootstrap weight. We use this block bootstrap as a default of the NeuBoots in sequel. Theoretically, the block bootstrap asymptotically converges to the non-blocked bootstrap as the number of blocks S increases as $n \to \infty$; see Theorem A.2 in the supplementray materials.

Training step Thanks to the block bootstrapping, the input of the resulting generator function G is $\boldsymbol{\alpha}$ of which dimension is reduced from n to S. Thus, we evaluate the generator by $G_\phi(X, \boldsymbol{\alpha}) = M_{\beta \oplus \beta_{\boldsymbol{\alpha}}} \circ (F_\theta(X) \oplus \boldsymbol{\alpha})$ and $\Phi[G_\phi]$ receives an input $w_{\boldsymbol{\alpha}}$. Note that $\nabla_\phi \Phi[G_\phi](w_{\boldsymbol{\alpha}}) = \langle w_{\boldsymbol{\alpha}}, \nabla_\phi L(G_\phi(\boldsymbol{\alpha}), \mathcal{D}_{\text{train}}) \rangle$. Then we can optimize equation 3.2 through the gradient descent by a Monte Carlo approximation. At every epoch, we randomly update the $w_{\boldsymbol{\alpha}}$, and the expectation

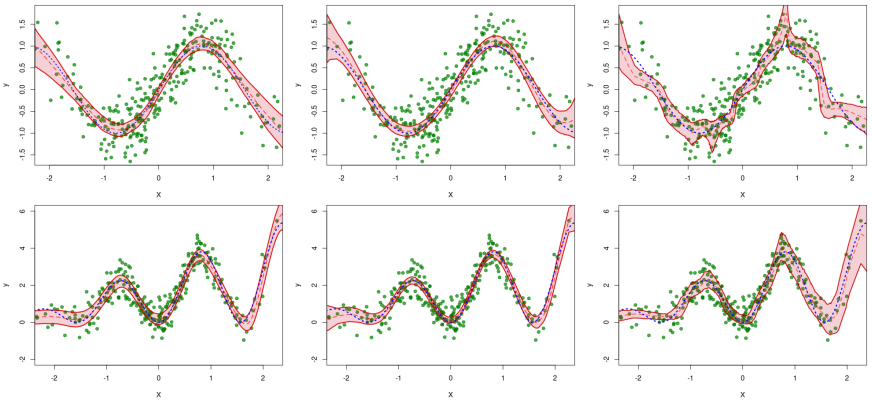

Figure 3.2: Two examples with different regression funtions. 95% confidence band of the regression mean from the NeuBoots with 10,000 bootstrap samples (the first column). 95% credible bands of the regression mean from the GP (the second column) and from the MCDrop (the third column). Each red dashed line indicates the mean, and the blue dotted lines show the true regression function.

in equation 3.2 can be approximated by the average over the sampled weights. Considering the stochastic gradient descent (SGD) algorithms to update the network parameter ϕ gradually via mini-batch sequence $\{\mathcal{D}_k : \mathcal{D}_k \subset \mathcal{D}_{\text{train}}\}_{k=1}^{K}$, we plug mini-batch size of $w_{\boldsymbol{\alpha},k} = \{\alpha_{u(i)} : X_i \in \mathcal{D}_k\}$ in equation 3.2 instead of full-batch size of $w_{\boldsymbol{\alpha}}$ without changing $\boldsymbol{\alpha}$. Note that each element of $w_{\boldsymbol{\alpha}}$ is not used repeatedly during the epoch, so the sampling and replacement procedures in Algorithm 1 are conducted once at the beginning of epoch.

Feature-Adaptive NeuBoots Modern neural networks can have different size of feature vector according to the data. For example, ResNet or DenseNet have a smaller size of the feature vector in CIFAR than ImageNet. In that case, a large block size S in NeuBoots can degrade the performance of the networks. Furthermore, fitting the hyperparamter S is another painful task. Hence we additionally propose *feature-adaptive* NeuBoots that removes hyperparameter S in the Algorithm 1. Recall the decomposition $f = M_\beta \circ F_\theta$ in Section 3.2. Then, instead of choosing an arbitrary number of blocks, we take S equals the dimension of the output of F_θ, and set $G_\phi(X, w_{\boldsymbol{\alpha}}) = M_\beta(F_\theta(X) \odot w_{\boldsymbol{\alpha}})$, where $\phi = (\theta, \beta)$ and $\odot$ denotes an elementwise multiplication. For a stable training, we utilize a simple heuristic *babysitting* i.e. initially, we train a model by setting $w_{\boldsymbol{\alpha}}$ as one vector for $t < T_{\text{BS}}$ epoch, and then we apply NeuBoots training. Consequently, these modifications show significant improvement in calibration and active learning (see Section 4.2 and 4.3).

Prediction step After training network G_ϕ, for the prediction, let a data point X_* be given and obtain the generator $\widehat{G}_*(\cdot) = G_\phi(X_*, \cdot)$. Then we can generate bootstrapped predictions by plugging $\boldsymbol{\alpha}^{(1)}, \ldots, \boldsymbol{\alpha}^{(B)}$ in the generator $\widehat{G}_*$, as described in Algorithm 2. Note that the algorithm evaluates the network from the scratch for only once to obtain the generator $\widehat{G}_*$, while the traditional bootstrapping needs repetitive feed-forward propagations. Hence it brings a computational advantage of the proposed method compared to Gal and Ghahramani (2016), which requires multiple numbers of feed-forward evaluations for the sampling of outputs. To check this empirically, we measure the inference time by ResNet-34 between NeuBoots and MCDrop on the test set of CIFAR-10 with V-100 GPUs. NeuBoots predicts $B = 100$ bootstrapping in 1.9s whereas MCDrop takes 112s for 100-times sampling.

3.4 Illustrative Example: Nonparametric Regression

To validate the empirical properties of the proposed method, we estimate 95% confidence band for nonparametric regression function by using the NeuBoots, and compare it with credible bands evaluated by Gaussian Process (GP) regression based on a radial basis kernel and MCDrop (Gal and Ghahramani, 2016). We use a MLP, which contains 3 hidden-layers with 500 hidden nodes for each layer, to model the generator of the NeuBoots. We adopt Algorithm 1 to train the neural net for 2000 epochs. Two illustrative examples are considered in Figure 3.2. The NeuBoots shows a

similar interval estimation with the GP, and more reliable confidence interval than MCDrop. We derive the confidence intervals with same number of samples, which shows NeuBoots can evaluate a valid confidence band, and the evaluated confidence band is comparable with the credible band of the Bayesian GP regression. Of course, the bootstrap distribution is not a posterior, so the interpretation of the confidence band cannot be the same with that of the Bayesian counterpart, but they surprisingly look similar. On the other hand, even though the MCDrop theoretically approximates the Bayesian GP regression, the resulting credible band is non-smooth and inconsistent with the shape of its target. At the end of the feature support, the credible band of the MCDrop is clearly narrower compared to the NeuBoots and the GP.

Furthermore, the NeuBoots is obviously scalable compared to the classical GP, since the conventional GP requires a $n \times n$ matrix inversion that demands $O(n^3)$ computational complexity, and its computation is practically infeasible for large-sized data sets. Instead, we compare the NeuBoots with a sparse approximation of the GP proposed by Snelson and Ghahramani (2006), and this approximated GP considers a small number, say m, of pseudo data points. Then, its computational complexity can be reduced to $O(m^3) + O(m^2 n)$, and we set $m = \sqrt{n}$. Figure 3.1 compares the computation times of the NeuBoots and the GP, and the results show that the NeuBoots is significantly faster than the sparse GP regression.

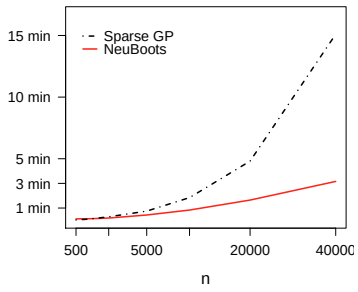

Figure 3.1: Comparison of computational time for the sparse GP and the NeuBoots.

4 Empirical Studies

In this section, we conduct the empirical studies of NeuBoots for uncertainty quantification and its applications. We apply NeuBoots to out-of-distribution experiments, confidence estimation, and active learning on the image classification tasks with deep convolutional neural networks.

4.1 Out-of-Distribution Detection Experiments

Setting As an important application of uncertainty quantification, we have applied NeuBoots to detection of out-of-distribution (OOD) samples. At first, we train ResNet-34 for the classification task in CIFAR-10 (in-distribution). We use the test datasets only for model evaluation. Then, we evaluate the performance of NeuBoots for OOD detection in the SVHN (out-of-distribution). For each model and dataset, we tune hyperparameters in the training phase using in-distribution samples to keep the fairness of our method. In the evaluation phase, we use a logistic regression based detector which outputs a confidence score for given test sample to discriminate OOD samples from in-distribution dataset. To evaluate the performance of the detector, we measure the true negative rate (TNR) at 95% true positive rate (TPR), the are under the receiver operating characteristic curve (AUROC), the area under the precision-recall curve (AUPR), and the detection accuracy. For comparison, we examine the baseline method (Hendrycks and Gimpel, 2017), ODIN (Liang et al., 2018), and Mahalanobis (Lee et al., 2018). For our method, we tune whole hyperparameters using a separate validation set, which consists of 1,000 images from in-distribution and out-distribution, respectively. After the training, we estimate the following four statistics regarding logit vectors: the max of predictive mean vectors, the standard deviation of logit vectors, expected entropy, and predictive entropy, which can be computed by the sampled output vectors of NeuBoots. Based on these statistics, similar to Ma et al. (2018); Lee et al. (2018), we tune the weights of logistic regression detector using nested cross-validation within the validation set, where the label is annotated positive for in-distribution sample and annotated negative for out-distribution sample.

Results Table 1 shows NeuBoots and feature-adaptive NeuBoots significantly outperform the baseline method Hendrycks and Gimpel (2017) and ODIN (Liang et al., 2018) without any calibration technique in OOD detection. Furthermore, with the input pre-processing technique studied in Liang et al. (2018), NeuBoots is superior to Mahalanobis (Lee et al., 2018) in almost metrics, which employs both the feature ensemble and the input pre-processing for the calibration techniques. This validates

Method	TNR at TPR 95%	AUROC	Detection Accuracy	AUPR In	AUPR Out
Baseline	32.47	89.88	85.06	85.4	93.96
ODIN	86.55	96.65	91.08	92.54	98.52
Mahalanobis	54.51	93.92	89.13	91.54	98.52
NeuBoots	**91.66**	97.18	**94.75**	**95.07**	98.54
NeuBoots-FA	89.40	**97.26**	93.80	93.97	**98.86**
Mahalanobis + Calibration	96.42	99.14	95.75	**98.26**	99.6
NeuBoots + Calibration	**99.00**	**99.14**	**96.52**	97.78	**99.68**

Table 1: The comparison NeuBoots and Baseline (Hendrycks and Gimpel, 2017), ODIN (Liang et al., 2018), and Mahalanobis (Lee et al., 2018) on OOD detection. NeuBoots-FA means feature-adaptive NeuBoots. We train ResNet-34 on CIFAR-10, and SVHN is used as OOD. All values are percantages and the best results are indicated in bold.

Data	Model	Metric	Baseline	MCDrop	NeuBoots-FA	NeuBoots-BS
CIFAR-10	ResNet-34	ACC	95.12	**95.18**	94.89	95.11
		ECE	3.21	3.33	**3.02**	3.86
	ResNet-110	ACC	94.11	94.07	93.36	**94.17**
		ECE	4.46	3.96	1.96	**1.69**
	DenseNet	ACC	94.87	**95.05**	93.98	94.92
		ECE	3.20	2.72	2.91	**2.16**
CIFAR-100	ResNet-34	ACC	77.88	78.33	77.54	**79**
		ECE	7.86	**7.45**	9.58	12.14
	ResNet-110	ACC	72.85	**73.63**	71.70	73.02
		ECE	16.58	14.89	**6.3**	8.0
	DenseNet	ACC	75.39	76.21	74.28	**76.43**
		ECE	12.67	9.12	**1.28**	3.82

Table 2: Comparison of the accuracy (ACC) and the ECE on CIFAR and architectures. All values are percantages and the best results are indicated in bold.

NeuBoots can discriminate OOD samples effectively. In order to see the performance change of the OOD detector concerning the bootstrap sample size, we evaluate the predictive standard deviation estimated by the proposed method for different $B \in \{2, 5, 10, 20, 30\}$. Figure B.1 illustrates that, for in-distribution classes (top row), NeuBoots predicts with extremely low uncertainty as expected. On the other hand, for out-distribution classes (bottom row), the proposed method predicts with increased uncertainty. As the number of bootstrap samples B increases, the predictive standard deviation for out-distribution classes increases, so that NeuBoots can detect OOD samples better.

4.2 Confidence Estimation via Feature-Adaptive NeuBoots

Setting We evaluate the proposed method on the confidence estimation with image classification task. We have applied feature-adaptive NeuBoots (NeuBoots-FA) and its babysitting version (NeuBoots-BS) to image classification tasks in CIFAR-10 and CIFAR-100 with ResNet-34, ResNet-110 and DenseNet. The size of bootstrap samples is $B = 100$ for prediction, and fix the other hyperparameters same with baseline models. All models are trained using SGD with a momentum of 0.9, an initial learning rate of 0.1, and a weight decay of 0.0005 with the mini-batch size of 128. We use CosineAnnealing for the learning rate scheduler. For NeuBoots-BS, we set $T_{\text{BS}} = 30$. We implement MCDrop and evaluates its performance with dropout rate $p = 0.2$, which is a close setting to the original paper. For the metric, we use the expected calibration error (ECE; Naeini et al. (2015)).

Results Table 2 validates that both NeuBoots-FA and NeuBoots-BS generally show better confidence estimation performances compared to baseline and MCDrop. These results show that NeuBoots is a reliable uncertainty quantification method encouraging a classifier to have robust predictions. Observe that NeuBoots-BS secures both accuracy and confidence estimation in the image classification tasks. Figure 4.1 shows the reliability diagrams and confidence histograms on CIFAR-100 and DenseNet. These plots demonstrate that NeuBoots significantly improve confidence estimation.

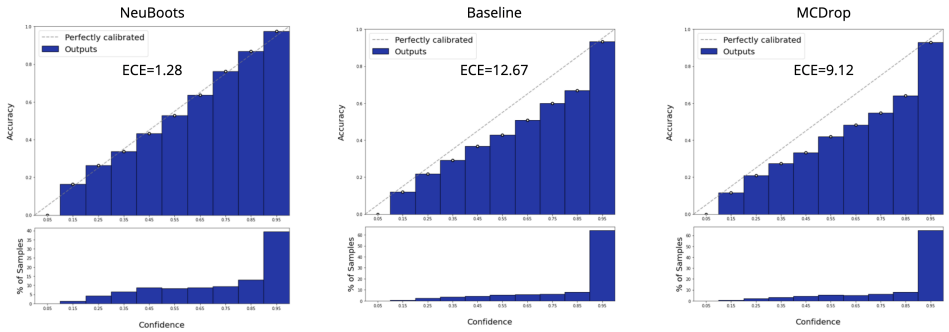

Figure 4.1: Comparison of reliability diagrams and confidence histograms on CIFAR-100 with DenseNet between feature-adaptive NeuBoots (left) baseline (middle) and MCDrop (right).

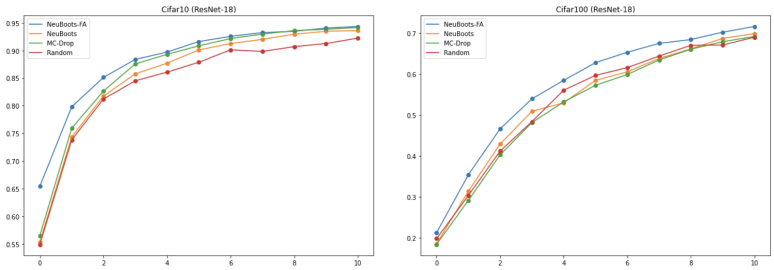

Figure 4.2: Actice learning performance on CIFAR-10 (left) and CIFAR-100 (right) with various sampling methods. Curves are averages over five runs.

4.3 Active Learning

Setting We evaluate the original NeuBoots and NeuBoots-FA on the active learning with ResNet-18 architecture on CIFAR-10 and CIFAR-100. For a comparison, we consider MCDrop with entropy-based sampling and random sampling. We follow an ordinary process to evaluate the performance of active learning (see Moon et al. (2020) for more details). Initially, a randomly sampled 2,000 labeled images are given, and we train a model. Based on the uncertainty estimation of each model, we sample 2,000 additional images from the unlabeled dataset and add to the labeled dataset for the next stage. We continue this process ten times for a single trial and repeat five trials for each model.

Results Figure 4.2 shows the sequential performance improvement on CIFAR-10 and CIFAR-100. Note that CIFAR-100 is more challenging dataset than CIFAR-10. Both plots demonstrate that NeuBoots-FA is superior to the other sampling methods in the active learning task. NeuBoots-FA records 71.6% accuracy in CIFAR-100 and 2.5% gap with MCDrop. Through the experiment, we verify that NeuBoots has a significant advantage in active learning.

5 Conclusion

We introduced a neural extension of bootstrap procedure, called the NeuBoots. While we applied the NeuBoots to OOD, confidence estimation, and active learning, the NeuBoots has an attractive potential for general purpose of neural net problems. The NeuBoots provides a valid confidence band of a regression function, and it can be considered as a potential alternative of the GP regression under the era of big data. One may extend the NeuBoots to recurrent neural networks or other architectures considering the domains such as text modeling, natural language processing, and reinforcement learning. We hope that the NeuBoots contributes to solving more challenging problems in future.

REFERENCES

Carlstein, E., Do, K.-A., Hall, P., Hesterberg, T., Künsch, H. R., et al. (1998). Matched-block bootstrap for dependent data. *Bernoulli*, 4(3):305–328.

CireşAn, D., Meier, U., Masci, J., and Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural networks*, 32:333–338.

Cortés-Ciriano, I. and Bender, A. (2018). Deep confidence: a computationally efficient framework for calculating reliable prediction errors for deep neural networks. *Journal of chemical information and modeling*, 59(3):1269–1281.

Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap methods and their application*, volume 1. Cambridge university press.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26.

Efron, B. (1987). Better bootstrap confidence intervals. *Journal of the American statistical Association*, 82(397):171–185.

Efron, B. (2000). The bootstrap and modern statistics. *Journal of the American Statistical Association*, 95(452):1293–1296.

Franke, J. and Neumann, M. H. (2000). Bootstrapping neural networks. *Neural computation*, 12(8):1929–1949.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356.

Hall, P. (1986). On the bootstrap and confidence intervals. *The Annals of Statistics*, pages 1431–1452.

Hall, P. (1992). On bootstrap confidence intervals in nonparametric regression. *The Annals of Statistics*, pages 695–711.

Hall, P. (1994). Methodology and theory for the bootstrap. *Handbook of econometrics*, 4:2341–2381.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hendrycks, D. and Gimpel, K. (2017). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *Proceedings of International Conference on Learning Representations*.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Kuleshov, V., Fenner, N., and Ermon, S. (2018). Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning*, pages 2796–2804.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, K., Lee, K., Lee, H., and Shin, J. (2018). A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177.

Liang, S., Li, Y., and Srikant, R. (2018). Enhancing the reliability of out-of-distribution image detection in neural networks. In *6th International Conference on Learning Representations, ICLR 2018*.

Louizos, C. and Welling, M. (2017). Multiplicative normalizing flows for variational bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2218–2227. JMLR. org.

Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S., Schoenebeck, G., Houle, M. E., Song, D., and Bailey, J. (2018). Characterizing adversarial subspaces using local intrinsic dimensionality. In *International Conference on Learning Representations*.

Moon, J., Kim, J., Shin, Y., and Hwang, S. (2020). Confidence-aware learning for deep neural networks. In *international conference on machine learning*.

Naeini, M. P., Cooper, G. F., and Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, volume 2015, page 2901. NIH Public Access.

Nalisnick, E. and Smyth, P. (2017). The amortized bootstrap. In *ICML Workshop on Implicit Models.*

Newton, M. A. and Raftery, A. E. (1994). Approximate Bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(1):3–26.

Præstgaard, J. and Wellner, J. A. (1993). Exchangeably weighted bootstraps of the general empirical process. *The Annals of Probability*, pages 2053–2086.

Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. (2014). Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*.

Rubin, D. B. (1981). The Bayesian bootstrap. *The Annals of Statistics*, 9(1):130434.

Shao, J. and Tu, D. (1996). *The jackknife and bootstrap*. Springer Science & Business Media.

Shin, M., Lee, Y., and Liu, J. S. (2020). Scalable uncertainty quantification via generative bootstrap sampler. *arXiv preprint arXiv:2006.00767*.

Smith, L. and Gal, Y. (2018). Understanding Measures of Uncertainty for Adversarial Example Detection. In *UAI*.

Snelson, E. and Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264.

A PROOF OF THEOREMS

In this section, we provide theoretical results in the main paper.

EXACTNESS OF NEUBOOTS

Theorem A.1. *Suppose that $\widehat{G}$ is the solution of equation 3.2. For each $\mathbf{w} \in \mathcal{W}$, set*

$$\widehat{f}_{\mathbf{w}} = \arg\min_{f \in \mathcal{N}} \Lambda[f](\mathbf{w}). \tag{A.1}$$

Then, for any probability distribution $\mathbb{P}_{\mathbf{w}}$ on $\mathcal{W}$, $\mathbb{E}_{\mathbf{w}}(\Phi[\widehat{G}](\mathbf{w})) = \mathbb{E}_{\mathbf{w}}(\Lambda[\widehat{f}_{\mathbf{w}}](\mathbf{w}))$. Furthermore, if the solution in equation 3.1 is unique, it holds that $\widehat{G}(X_i, \mathbf{w}) = \widehat{f}_{\mathbf{w}}(X_i)$ almost surely for $i = 1, \ldots, n$.

The unique solution condition, assumed in Theorem A.1, is somewhat strong in practice, because a large-sized neural network is over-parameterized and has multiple solutions of the loss function. However, the NeuBoots successfully evaluates the bootstrapped neural networks in various empirical examples that we examined in Section 4.

Proof. Note that $G(\mathbf{w}, \cdot) \in \mathcal{N}$ for fixed $\mathbf{w} \in \mathcal{W}$ and $\widehat{f}_{\mathbf{w}}$ is determined by equation A.1 for given $\mathbf{w} \in \mathcal{W}$ hence $\widehat{f}_{\cdot} \in \mathcal{G}$. Due to equation A.1, we have

$$\Lambda[\widehat{f}_{\mathbf{w}}](\mathbf{w}) \leq \Phi[G](\mathbf{w}),$$

for each $G \in \mathcal{G}$. This means that, for a given $\mathbb{P}_{\mathbf{w}}$, it holds

$$\mathbb{E}_{\mathbf{w}}(\Lambda[\widehat{f}_{\mathbf{w}}](\mathbf{w})) \leq \mathbb{E}_{\mathbf{w}}(\Phi[G](\mathbf{w})), \quad \forall G \in \mathcal{G}. \tag{A.2}$$

Also, by the definition of $\widehat{G}$, we have

$$\mathbb{E}_{\mathbf{w}}(\Phi[\widehat{G}](\mathbf{w})) \leq \mathbb{E}_{\mathbf{w}}(\Phi[\widehat{f}_{\mathbf{w}}](\mathbf{w})) \text{ a.s.} \tag{A.3}$$

Combining equation A.2 and equation A.3, the theorem is proved. □

ASYMPTOTICS OF BLOCK BOOTSTRAP

We shall rigorously investigate asymptotic equivalence between the blocked bootstrap and the non-blocked bootstrap. To ease the explanation for theory, we introduce some notation here. We distinguish a random variable Y_i and its observed value y_i, and we assume that the feature $X_1, X_2, \ldots$ is deterministic. the Euclidean norm is denoted by $\|\cdot\|$, and the norm of a L_2 space is denoted by $\|\cdot\|_2$. Also, to emphasize that the bootstrap weight $\mathbf{w}$ depends on n, we use $\mathbf{w}_n$. Let $Y_1, Y_2, \ldots$ be i.i.d. random variables from the probability measure space $(\Omega, \mathcal{F}, \mathbb{P}_0)$. We denote the empirical probability measure by $\widehat{\mathbb{P}}_n := \sum_{i=1}^n \delta_{Y_i}/n$, where δ_x is a discrete point mass at $x \in \mathbb{R}$, and let $\mathbb{P}g = \int g d\mathbb{P}$, where $\mathbb{P}$ is a probability measure and g is a $\mathbb{P}$-measurable function. Suppose that $\sqrt{n}(\widehat{\mathbb{P}}_n - \mathbb{P}_0)$ weakly converges to a probability measure $\mathbb{T}$ defined on some sample space and its sigma field $(\Omega', \mathcal{F}')$. In the regime of bootstrap, what we are interested in is to estimate $\mathbb{T}$ by using some weighted empirical distribution that is $\widehat{\mathbb{P}}_n^* = \sum_{i=1}^n w_i \delta_{Y_i}$, where $w_1, w_2, \ldots$ is an i.i.d. weight random variable from a probability measure $\mathbb{P}_{\mathbf{w}}$. In the same sense, the probability measure acts on the block bootstrap is denoted by $\mathbb{P}_{\mathbf{w}_\alpha}$. We state a primary condition on bootstrap theory as follows:

$$\sqrt{n}(\widehat{\mathbb{P}}_n g - \mathbb{P}_0 g) \to \mathbb{T}g \quad \text{for } g \in \mathcal{D} \quad \text{and} \quad \mathbb{P}_0 g_{\mathcal{D}}^2 < \infty, \tag{A.4}$$

where $\mathcal{D}$ is a collection of some continuous functions of interest, and $g_{\mathcal{D}}(\omega) = \sup_{g \in \mathcal{D}} |g(\omega)|$ is the envelope function on $\mathcal{D}$. This condition means that there exists a target probability measure and the functions of interest should be square-bounded.

Based on this condition, the following theorem states that the block bootstrap asymptotically induces the same bootstrap distribution with that of non-block bootstrap. All proofs of theorems are deferred to the supplementary material.

Theorem A.2. *Suppose that equation A.4 holds and* $\{\alpha_1, \ldots, \alpha_S\}^{\mathrm{T}} \sim S \times Dirchlet(1, \ldots, 1)$ *with* $w_i = \alpha_{u(i)}$. *We assume some regularity conditions introduced in the supplementary material, and also assume* $S \to \infty$ *as* $n \to \infty$. *Then, for a* r_n *such that* $\|\widehat{f} - f_0\|_2 = O_{\mathbb{P}_{\mathbf{w}}}(\zeta_n r_n^{-1})$ *for any diverging sequence* ζ_n,

$$\sup_{x \in \mathcal{X}, U \in \mathcal{B}} \left| \mathbb{P}_{\mathbf{w}} \left\{ r_n(\widehat{f}_{\mathbf{w}}(x) - \widehat{f}(x)) \in U \right\} - \mathbb{P}_{\mathbf{w}_{\boldsymbol{\alpha}}} \left\{ r_n(\widehat{f}_{\mathbf{w}_{\boldsymbol{\alpha}}}(x) - \widehat{f}(x)) \in U \right\} \right| \to 0, \tag{A.5}$$

in $\mathbb{P}_0$*-probability, where* $\mathcal{B}$ *is the Borel sigma algebra.*

Recall that the notation is introduced in Section 3.3.

Præstgaard and Wellner (1993) showed that the following conditions on the weight distribution to derive bootstrap consistency for general settings:

W1. $\mathbf{w}_n$ is exchangeable for $n = 1, 2, \ldots$.
W2. $w_{n,i} \geq 0$ and $\sum_{i=1}^n w_{n,i} = n$ for all n.
W3. $\sup_n \|w_{n,1}\|_{2,1} < \infty$, where $\|w_{n,1}\|_{2,1} = \int \sqrt{\mathbb{P}_{\mathbf{w}}(w_{n,1} \geq t)} dt$.
W4. $\lim_{\lambda \to \infty} \limsup_{n \to \infty} \sup_{t \geq \lambda} t^2 \mathbb{P}_{\mathbf{w}}(w_{n,1} \geq t) = 0$.
W5. $n^{-1} \sum_{i=1}^n (w_{n,i} - 1)^2 \to 1$ in probability.

Under **W1**-**W5**, combined with equation A.4, showed that $\sqrt{n}(\widehat{\mathbb{P}}_n^* - \widehat{\mathbb{P}}_n)$ weakly converges to $\mathbb{T}$. It was proven that the Dirichlet weight distribution satisfies **W1**-**W5**, and we first show that the Dirichlet weight distribution for the blocks also satisfies the condition. Then, the block bootstrap of the empirical process is also consistent when the classical bootstrap of the empirical process is consistent.

Since the block bootstrap randomly assigns subgroups, the distribution of $\mathbf{w}_n$ is exchangeable, so the condition **W1** is satisfied. The condition **W2** and **W3** are trivial. Since a Dirichlet distribution with a unit constant parameter can be approximated by a pair of independent exponential random variables; i.e $\{z_1 / \sum_{i=1}^S z_i, \ldots, z_S / \sum_{i=1}^S z_i\} \sim Dir(1, \ldots, 1)$, where $z_i \overset{i.i.d.}{\sim} \exp(1)$. Therefore, $S \times Dir(1, \ldots, 1) \approx \{z_1, \ldots, z_S\}$, if S is large enough. This fact shows that $t^2 \mathbb{P}_{\mathbf{w}}(w_{n,1} \geq t) \approx t^2 \mathbb{P}_z(z_1 \geq t)$, and it follows that $\mathbb{P}_z(z_1 \geq t) = \exp(-t)$, so **W4** is shown. The condition **W5** is trivial by the law of large number. Then, under **W1**-**W5**, Theorem 2.1 in Præstgaard and Wellner (1993) proves that

$$\sqrt{n}(\widehat{\mathbb{P}}_n^* - \widehat{\mathbb{P}}_n) \Rightarrow \mathbb{T}, \tag{A.6}$$

where the convergence "$\Rightarrow$" indicates weakly convergence.

We denote the true neural net parameter by ϕ_0 such that $f_0 = f_{\phi_0}$, where f_0 is the true function that involves in the data generating process, and $\widehat{\phi}$ and $\widehat{\phi}_{\mathbf{w}}$ are the minimizers of the equation 3.1 for one-vector (i.e. $\mathbf{w} = (1, \ldots, 1)$) and given $\mathbf{w}$, respectively. This indicates that $\widehat{f} = f_{\widehat{\phi}}$ and $\widehat{f}_{\mathbf{w}} = f_{\widehat{\phi}_{\mathbf{w}}}$. Then, our objective function can be expressed as minimizing $\widehat{P}_n L(f_\phi(X), y)$ with respect to ϕ. We further assume that

A1. the true function belongs to the class of neural network, i.e. $f_0 \in \mathcal{F}$.

A2. $\sup_{x \in \mathcal{X}, U \in \mathcal{B}} \left| \mathbb{P}_{\mathbf{w}} \left\{ r_n(\widehat{f}_{\mathbf{w}}(x) - \widehat{f}(x)) \in U \right\} - \mathbb{P}_0 \left\{ r_n(\widehat{f}(x) - f_0(x)) \in U \right\} \right| \to 0,$

in $\mathbb{P}_0$-probability, where f_0 is the true function that involves in the data generating process.

A3. Suppose that $\sum_{i=1}^n \frac{\partial}{\partial \phi} L(f_{\widehat{\phi}}(X_i), y_i) = 0$, $\sum_{i=1}^n \frac{\partial}{\partial \phi} w_i L(f_{\widehat{\phi}_{\mathbf{w}}}(X_i), y_i) = 0$ for any $\mathbf{w}$, and $\mathbb{E}_0[\frac{\partial}{\partial \phi} L(f_{\phi_0}(X), y)] = 0$.

A4. $\mathcal{H}$ is in $\mathbb{P}_0$-Donsker family, where $\mathcal{H} = \{\frac{\partial}{\partial \phi} L(f_\phi(\cdot), \cdot) : \phi \in \Phi\}$; i.e. $\sqrt{n}(\widehat{\mathbb{P}}_n g - \widehat{\mathbb{P}}_0 g) \to \mathbb{T} g$ for $g \in \mathcal{H}$ and $\mathbb{P}_0 g_{\mathcal{H}}^2 < \infty$.

These conditions assume that the classical weighted bootstrap is consistent, and a rigorous theoretical investigation of this consistency is non-existent at the current moment. However, we remark that the main purpose of this theorem is to show that the considered block bootstrap induces asymptotically the same result from the classical non-block bootstrap so that the use of the block bootstrap is at least

asymptotically equivalent to the classical counterpart. In this sense, it is reasonable to assume that the classical bootstrap is consistent.

Then, it follows that

$$\begin{aligned}
& \sup_{x\in\mathcal{X},U\in\mathcal{B}} \left|\mathbb{P}_{\mathbf{w}}\left\{r_n(\widehat{f}_{\mathbf{w}}(x)-\widehat{f}(x))\in U\right\}-\mathbb{P}_{\mathbf{w}_\alpha}\left\{r_n(\widehat{f}_{\mathbf{w}_\alpha}(x)-\widehat{f}(x))\in U\right\}\right| \\
\leq & \sup_{x\in\mathcal{X},U\in\mathcal{B}} \left|\mathbb{P}_{\mathbf{w}}\left\{r_n(\widehat{f}_{\mathbf{w}}(x)-\widehat{f}(x))\in U\right\}-\mathbb{P}_0\left\{r_n(\widehat{f}(x)-f_0(x))\in U\right\}\right| \\
& + \sup_{x\in\mathcal{X},U\in\mathcal{B}} \left|\mathbb{P}_{\mathbf{w}_\alpha}\left\{r_n(\widehat{f}_{\mathbf{w}_\alpha}(x)-\widehat{f}(x))\in U\right\}-\mathbb{P}_0\left\{r_n(\widehat{f}(x)-f_0(x))\in U\right\}\right|.
\end{aligned}$$

The first part in the right-hand side of the inequality converges to 0 by **A1**. Also, the second part also converges to 0. That is because the empirical process of the block weighted bootstrap is asymptotically equivalent to the classical RWB, so **A2** and **A3** guarantees that the asymptotic behavior of the bootstrap solution should be consistent as the classical counterpart does. □

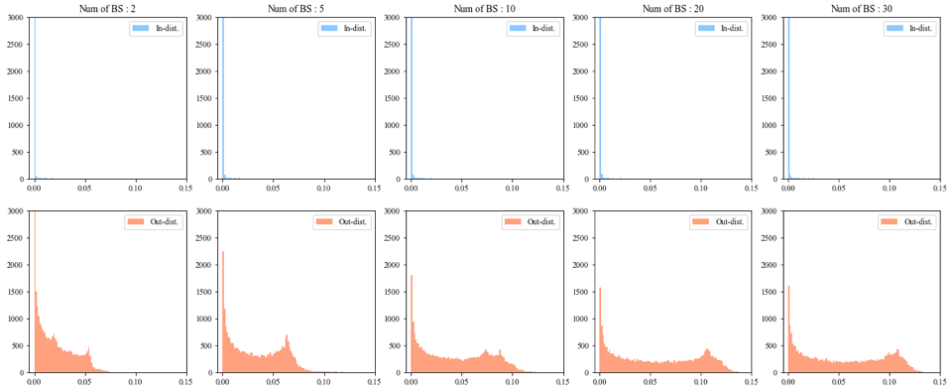

Figure B.1: Histogram of the predictive standard deviation estimated by NeuBoots on test samples from CIFAR-10 (in-distribution) classes (top row) and SVHN (out-distribution) classes (bottom row), as we vary bootstrap sample size $B \in \{2, 5, 10, 20, 30\}$.

B ADDITIONAL EXPERIMENTAL RESULTS

In this section, we illustrate additional results of OOD detection experiments.

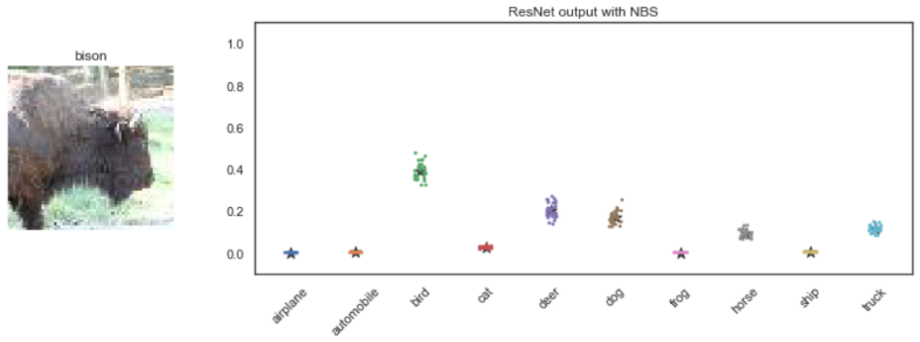

Figure B.2: Confidence bands of the prediction of NeuBoots for `bison` data in TinyImageNet. The proposed method predicts is as an out-of-distribution class with `prob`=0.7664.

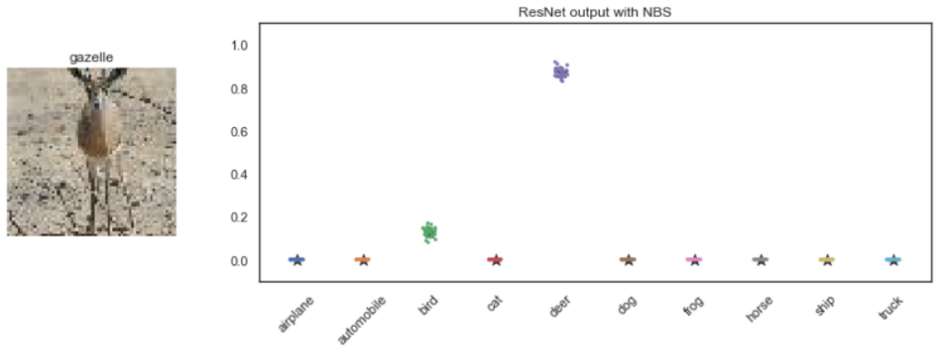

Figure B.3: Confidence bands of the prediction of NeuBoots for `gazelle` data in TinyImageNet. The proposed method predicts is as an out-of-distribution class with `prob`=0.5021.

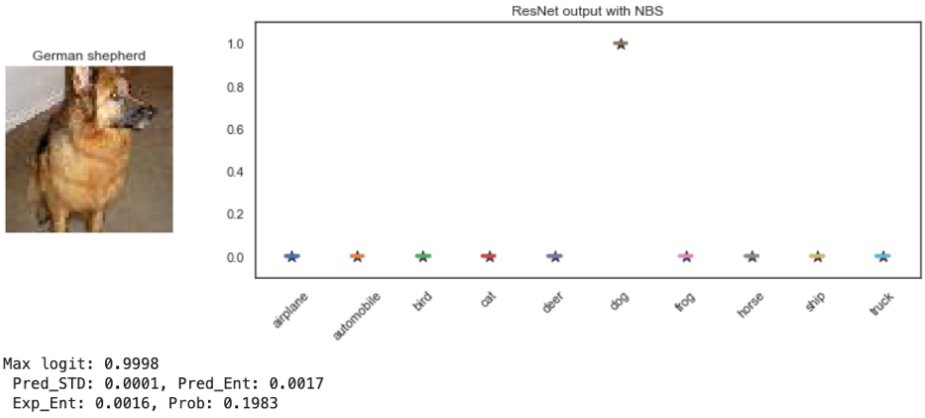

Figure B.4: Confidence bands of the prediction of NeuBoots for `German shepherd` data in TinyImageNet. The proposed method predicts is as an in-of-distribution class `dog` with `prob`=0.1983.