Accuracy-Preserving Calibration via Statistical Modeling on Probability Simplex

Yasushi Esaki[†*] **Akihiro Nakamura**[†] **Keisuke Kawano**[†]

Ryoko Tokuhisa[†] **Takuro Kutsuna**[†]

[†]Toyota Central R&D Labs., Inc.

Abstract

Classification models based on deep neural networks (DNNs) must be calibrated to measure the reliability of predictions. Some recent calibration methods have employed a probabilistic model on the probability simplex. However, these calibration methods cannot preserve the accuracy of pre-trained models, even those with a high classification accuracy. We propose an accuracy-preserving calibration method using the Concrete distribution as the probabilistic model on the probability simplex. We theoretically prove that a DNN model trained on cross-entropy loss has optimality as the parameter of the Concrete distribution. We also propose an efficient method that synthetically generates samples for training probabilistic models on the probability simplex. We demonstrate that the proposed method can outperform previous methods in accuracy-preserving calibration tasks using benchmarks.

1 INTRODUCTION

To ensure the safety of systems operated by deep neural network (DNN) classification models, the reliability of the model predictions must be measured for each input. If the confidence (Guo et al., 2017) computed by the models matches the classification accuracy, we can confidently measure the reliability of the predictions for unlabeled inputs. Many previous methods train the models until the confidence matches the accuracy (Zou et al., 2019; Wilson and Izmailov, 2020). Such a training scheme is called calibration (Guo et al., 2017). However, models trained under calibration tend to have lower accuracy than models trained only for classification accuracy (Thulasidasan et al., 2019). Degrading the accuracy to improve the model calibration is undesired. In this paper, we instead aim to calibrate the confidence of a highly accurate model trained without considering calibration while preserving its accuracy. This approach is called accuracy-preserving calibration (Zhang et al., 2020).

While many previous researches on calibration have adjusted the input-dependent parameter of the categorical distribution (Filho et al., 2021), some existing studies optimize the input-dependent parameter of the Dirichlet distribution for calibration (Malinin et al., 2020; Ryabinin et al., 2021). A probabilistic model on the probability simplex, such as the Dirichlet distribution, enables us to distinguish between two types of prediction uncertainty, aleatoric uncertainty and epistemic uncertainty (Hüllermeier and Waegeman, 2021), which cannot be distinguished by the categorical distribution. This advantage prevents overconfidence, which is caused by failure to estimate the epistemic uncertainty (Kristiadi et al., 2020). However, the above methods (Malinin et al., 2020; Ryabinin et al., 2021) cannot preserve the model accuracy because a DNN-based classification model is updated for optimizing the input-dependent parameter of the Dirichlet distribution. In addition, these methods obtain labels on the probability simplex from ensemble models to estimate the Dirichlet distribution. Since ensemble models comprise training a few tens of DNN models initialized with different random weight parameters, the training overhead is excessively large.

The existing accuracy-preserving calibration methods, such as Temperature Scaling (TS) (Guo et al., 2017),

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. *email: yasushi.esaki.sb@mosk.tytlabs.co.jp

introduce a temperature parameter separately from the logits of a pre-trained DNN model for the confidence computation. The temperature parameter is optimized to improve confidence while freezing the pre-trained DNN model. However, TS only adjusts the parameter of the categorical distribution and does not optimize the parameter of the probabilistic model on the probability simplex. Although many extensions to the original TS method (Zhang et al., 2020; Tomani et al., 2022; Joy et al., 2023) have been proposed, these methods also adjust the parameter of the categorical distribution. Therefore, the previous TS methods have difficulty distinguishing between the aleatoric uncertainty and the epistemic uncertainty, which limits the calibration performance.

Therefore, an accuracy-preserving calibration method that estimates a probabilistic model on the probability simplex is required. One possible approach is to combine TS and the Dirichlet distribution.[1] However, this approach has problems in optimizing the temperature parameter alone (see Section 4.2) and shows low calibration performance in our experiments (see Section 5.5).

In this paper, we propose Simplex TS (STS), which applies the Concrete distribution (Maddison et al., 2017) as the probabilistic model on the probability simplex. The Concrete distribution has two parameters, the location parameter and the temperature parameter. STS optimizes these two parameters in turn, corresponding to the training for classification and the subsequent accuracy-preserving calibration, respectively. More precisely, we prove theoretically that a DNN model trained with cross-entropy loss optimizes the location parameter independently of the temperature parameter. This fact allows us to reuse the pre-trained DNN model for optimizing the location parameter and optimize the temperature parameter solely for calibration. Both parameters of the Concrete distribution are optimized by the accuracy-preserving calibration alone, if we already have the pre-trained DNN model. To optimize the temperature parameter, we also propose Multi-Mixup, which synthetically generates training samples labeled with vectors on the probability simplex. Multi-Mixup does not have to train ensemble models and ameliorates the training overhead. Through numerical experiments, we demonstrate that STS can outperform previous TS methods in accuracy-preserving calibration tasks.

Our contributions are summarized below.

- We propose an accuracy-preserving calibration method using the Concrete distribution as a probabilistic model on the probability simplex (Section 4) and demonstrate that the proposed method outperforms previous methods (Section 5).
- We propose a method that synthetically generates a dataset for training probabilistic models on the probability simplex. This method reduces the training overhead from those of the previous ensemble methods (Section 4.3.1).

[1] An example is presented in Appendix D.

2 RELATED WORK

2.1 Accuracy-Preserving Calibration

TS (Guo et al., 2017) is a typical accuracy-preserving calibration method. As mentioned above, researchers have proposed many successor methods (Zhang et al., 2020; Balanya et al., 2022) that extend the original TS method (Guo et al., 2017). Some recent TS methods attempt to optimize the temperature parameter for each sample with a function that depends on the inputs (Tomani et al., 2022; Joy et al., 2023). The commonality of these TS methods is that they improve confidence by tuning a scalar-valued temperature parameter separately from a pre-trained DNN model. Accuracy-preserving calibration methods other than TS are also available (Kuleshov and Deshpande, 2022; Wenger et al., 2020; Ramalho and Miranda, 2020). Kuleshov and Deshpande (2022) train an auxiliary model to transform the softmax outputs of a pre-trained model for calibration and Wenger et al. (2020) optimize the parameter of the categorical distribution following the Gaussian process. These studies do not consider a probabilistic model on the probability simplex. Ramalho and Miranda (2020) aim to perform out-of-distribution detection while preserving a pre-trained DNN model, which deviates from the calibration in terms of problem settings.

2.2 Uncertainty Estimation on the Probability Simplex

Many studies estimate the prediction uncertainty, which consists of aleatoric uncertainty and epistemic uncertainty (Hüllermeier and Waegeman, 2021), using the Dirichlet distribution whose parameter is computed by an input-dependent DNN model (Malinin and Gales, 2018; Sensoy et al., 2018). Prior Networks (Malinin and Gales, 2018) are trained by minimizing the Kullback–Leibler divergence of the Dirichlet distribution. Evidential Deep Learning (Sensoy et al., 2018) measures the amount of evidence of each class by estimating the Dirichlet distribution. Uncertainty estimation methods that use a probabilistic model on the probability simplex, such as the Dirichlet distribution, can estimate aleatoric and epistemic

uncertainties distinctly. Note that the above studies do not discuss accuracy-preserving calibration, and to the best of our knowledge, no studies have applied a probabilistic model on the probability simplex to the accuracy-preserving calibration.

2.3 Data Augmentation by Interpolation

Some data augmentation methods generate pseudo-new samples by linearly interpolating the original samples (Zhang et al., 2018; Thulasidasan et al., 2019). Whereas many studies interpolate two samples (Inoue, 2018; Verma et al., 2019), AdaMixup (Guo et al., 2019) and ζ-Mixup (Abhishek et al., 2022) interpolate three or more samples, which is the same as Multi-Mixup. AdaMixup and ζ-Mixup aim to improve the classification performance and have mechanisms to restrict the generation of samples that differ significantly from the original samples. This property is not suitable for calibration because the features of inputs with a large prediction uncertainty cannot be learned.

3 PRELIMINARIES

3.1 Notations

We write a bold symbol $\boldsymbol{a}$ for the real vector, and let a_i be the i-th entry of $\boldsymbol{a}$ with $i \in \mathbb{N}$. The real vector-valued function is also written as a bold symbol $\boldsymbol{f}$, and let $f_i(\boldsymbol{x})$ be the i-th coordinate of the output of $\boldsymbol{f}(\boldsymbol{x})$. Furthermore, let $\mathbb{1}_{\{\cdot\}}$ be an indicator function and Δ_{q-1} be the $q-1$ dimensional probability simplex, where $q \in \mathbb{N}$ is an integer. Specifically, $\Delta_{q-1} := \{\boldsymbol{\pi} \in [0,1]^q | \sum_{i=1}^{q} \pi_i = 1\}$. Let $\mathcal{X} (\subseteq \mathbb{R}^d)$ be an input space and let $\mathcal{Y} = \{1, \dots, K\}$ be a set of class categories, where $K \in \mathbb{N}$ is the number of classes. Let $\mathrm{Cat}(y|\boldsymbol{\pi})$ be the probability mass function of the categorical distribution with a parameter $\boldsymbol{\pi} \in \Delta_{K-1}$. Let $\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\mu})$ be the probability density function of the Dirichlet distribution with a parameter $\boldsymbol{\mu} \in (0, \infty)^q$.

3.2 Uncertainty Estimation on the Probability Simplex

Malinin and Gales (2018) propose to estimate the prediction uncertainty in classification through a probabilistic model on the probability simplex. Let the class label $y \in \mathcal{Y}$ be a stochastic variable following a probability distribution $p(y|\boldsymbol{\pi})$, where $\boldsymbol{\pi} \in \Delta_{K-1}$ is a parameter. Let $\boldsymbol{\pi}$ be another stochastic variable following the conditional probability distribution $p(\boldsymbol{\pi}|\boldsymbol{x})$, where $\boldsymbol{x} \in \mathcal{X}$ is the input. A conditional class distribution $p(y|\boldsymbol{x})$ given an input $\boldsymbol{x}$ can be described as

$$p(y|\boldsymbol{x}) = \int p(y|\boldsymbol{\pi}) p(\boldsymbol{\pi}|\boldsymbol{x}) d\boldsymbol{\pi}. \tag{1}$$

To estimate the prediction uncertainty, Malinin and Gales (2018) estimate $p(\boldsymbol{\pi}|\boldsymbol{x})$ following the Dirichlet distribution. Let $\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\theta}_\mu) = (e^{g_1(\boldsymbol{x})}, \dots, e^{g_K(\boldsymbol{x})})^\top$, where $\boldsymbol{g} : \mathcal{X} \to \mathbb{R}^K$ is a DNN model (a model with logit outputs $\boldsymbol{g}(\boldsymbol{x})$) parameterized by a real vector $\boldsymbol{\theta}_\mu$. Malinin and Gales (2018) assume that y follows the categorical distribution with a parameter $\boldsymbol{\pi}$, whereas $\boldsymbol{\pi}$ follows the Dirichlet distribution with a $\boldsymbol{x}$-dependent parameter $\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\theta}_\mu)$:

$$\begin{aligned} p(y|\boldsymbol{\pi}) &= \mathrm{Cat}(y|\boldsymbol{\pi}), \\ p(\boldsymbol{\pi}|\boldsymbol{x}) &= \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\theta}_\mu)). \end{aligned} \tag{2}$$

If the estimated value of $\boldsymbol{\theta}_\mu$ is denoted by $\hat{\boldsymbol{\theta}}_\mu$, the prediction uncertainty is represented by the estimated distribution $\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\mu}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\mu))$, and we can estimate the prediction uncertainty for unlabeled inputs. The prediction uncertainty is estimated to be small when $\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\mu}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\mu))$ has a high probability density at one of the apexes of Δ_{K-1}. Conversely, the prediction uncertainty is estimated to be large when $\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\mu}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\mu))$ has a high probability density at the center of Δ_{K-1} or has a large variance over Δ_{K-1}. Specifically, a high probability density at the center of Δ_{K-1} corresponds to high aleatoric uncertainty, and a large variance over Δ_{K-1} corresponds to high epistemic uncertainty (Malinin and Gales, 2018). Applying a probabilistic model on Δ_{K-1} such as $\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\theta}_\mu))$ provides the distinction between aleatoric uncertainty and epistemic uncertainty, which cannot be distinguished when considering only the categorical distribution. This distinction helps confidence take a lower value when the number of training samples is small, corresponding to epistemic uncertainty, and avoids overconfidence (Kristiadi et al., 2020).

The previous studies discussing uncertainty estimation (Malinin et al., 2020; Lindqvist et al., 2020; Ryabinin et al., 2021) assume that the distribution of the softmax outputs of ensemble models input with $\boldsymbol{x}$ indicates the prediction uncertainty of $\boldsymbol{x}$, and estimate $\boldsymbol{\theta}_\mu$ on a dataset labeled with these softmax outputs to learn their distribution. The Dirichlet distribution is appropriate for learning the prediction uncertainty through the ensemble models because the softmax outputs of the ensemble models are vectors on Δ_{K-1} and their distribution is represented by a probabilistic model on Δ_{K-1}. Although these studies achieve high calibration performance (Malinin et al., 2020; Lindqvist et al., 2020; Ryabinin et al., 2021), they have difficulty in performing accuracy-preserving calibration.

3.3 Concrete Distribution

The Concrete distribution (Maddison et al., 2017) is a family of probability distributions on the probability

simplex. It is formulated by extending the categorical distribution to a continuous probability distribution and is parameterized by two parameters: the location parameter $\boldsymbol{\alpha} \in (0, \infty)^q$ and the temperature parameter $\lambda \in (0, \infty)$. The probability density function of the Concrete distribution is defined as follows.

Definition (Concrete distribution; (Maddison et al., 2017))**.** Let $\boldsymbol{\alpha} \in (0, \infty)^q$ and $\lambda \in (0, \infty)$. The probability density of a stochastic variable $\boldsymbol{\pi} \in \Delta_{q-1}$ following the Concrete distribution with location parameter $\boldsymbol{\alpha}$ and temperature parameter λ is given by

$$\mathrm{Cn}(\boldsymbol{\pi}|\boldsymbol{\alpha}, \lambda) \coloneqq (q-1)!\lambda^{q-1} \prod_{j=1}^{q} \left(\frac{\alpha_j \pi_j^{-(\lambda+1)}}{\sum_{i=1}^{q} \alpha_i \pi_i^{-\lambda}} \right). \quad (3)$$

The location parameter $\boldsymbol{\alpha}$ indicates the relative mass among the apexes of Δ_{q-1}, and the temperature parameter λ indicates the difference from the categorical distribution whose parameter is given by $(\alpha_1 / \sum_{i=1}^{q} \alpha_i, \ldots, \alpha_q / \sum_{i=1}^{q} \alpha_i)^\top$. The Concrete distribution has been adopted as the reparameterization trick in a variational autoencoder (Maddison et al., 2017; Jang et al., 2017). To the best of our knowledge, no research has used this distribution for calibration purposes.

Remark 1 (Properties of the temperature parameter)**.** The variance over Δ_{K-1} can be changed by shifting λ. Moreover, the probability density is concentrated at the apexes of Δ_{q-1} if λ is sufficiently small and at the center of Δ_{q-1} if λ is sufficiently large (Maddison et al., 2017). Note that this variation is realized even when $\boldsymbol{\alpha}$ is fixed. This property of λ implies that optimizing λ on a dataset that has labels indicating the prediction uncertainty contributes to uncertainty estimation.

4 SIMPLEX TEMPERATURE SCALING

In this section, we propose STS for accuracy-preserving calibration. We first present the overview of STS in Section 4.1. We then show theoretically that STS enables accuracy-preserving calibration in Section 4.2. The training of STS with the data generation method is described in Section 4.3. Section 4.4 covers the confidence calculation, while the implementation of STS is shown in Section 4.5.

4.1 Overview

This subsection introduces a probabilistic model and its optimization process for STS.

Probabilistic model. Let $\boldsymbol{\alpha}(\cdot, \boldsymbol{\theta}_\alpha) : \mathcal{X} \to (0, \infty)^K$ and $\lambda(\cdot, \boldsymbol{\theta}_\lambda) : \mathcal{X} \to (0, \infty)$ be deterministic functions parameterized by real vectors $\boldsymbol{\theta}_\alpha$ and $\boldsymbol{\theta}_\lambda$, respectively. Our probabilistic models for $p(y|\boldsymbol{\pi})$ and $p(\boldsymbol{\pi}|\boldsymbol{x})$ are respectively formulated as

$$p(y = k|\boldsymbol{\pi}) = \mathbb{1}_{\{\arg\max_{i \in \mathcal{Y}} \pi_i = k\}} \quad \text{for } \forall k \in \mathcal{Y}, \quad (4)$$

$$p(\boldsymbol{\pi}|\boldsymbol{x}) = \mathrm{Cn}(\boldsymbol{\pi}|\boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{\theta}_\alpha), \lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)). \quad (5)$$

As indicated in Eq. (4), $p(y|\boldsymbol{\pi})$ is given by a one-hot vector in which one component is 1 and the other components are 0. That is, the value of y is definitively determined from $\boldsymbol{\pi}$, meaning that y is a stochastic variable if and only if $\boldsymbol{\pi}$ is a stochastic variable. Although Eq. (4) has no variance, the combination of Eqs. (4) and (5) has good properties for consistently formulating the pre-training for classification and the subsequent accuracy-preserving calibration.

Optimization process. The pre-training for classification and the subsequent accuracy-preserving calibration can be formulated as consecutively estimating $\boldsymbol{\theta}_\alpha$ and $\boldsymbol{\theta}_\lambda$ using different criteria. Given a dataset $\mathcal{D} \coloneqq \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, where $y_n \in \mathcal{Y}$ is the class label of $\boldsymbol{x}_n \in \mathcal{X}$, STS performs the following two steps.

Step 1: Classification Based on $\mathcal{D}$, we estimate $\boldsymbol{\theta}_\alpha$ via maximum likelihood estimation of Eq. (1) under Eqs. (4) and (5). The estimated value of $\boldsymbol{\theta}_\alpha$ is denoted by $\hat{\boldsymbol{\theta}}_\alpha$.

Step 2: Calibration Based on $\tilde{\mathcal{D}}$, we estimate $\boldsymbol{\theta}_\lambda$ via maximum likelihood estimation of Eq. (5), fixing $\boldsymbol{\theta}_\alpha$ to $\hat{\boldsymbol{\theta}}_\alpha$. The estimated value of $\boldsymbol{\theta}_\lambda$ is denoted by $\hat{\boldsymbol{\theta}}_\lambda$.

In Step 2, $\tilde{\mathcal{D}} (\coloneqq \{(\tilde{\boldsymbol{x}}_m, \tilde{\boldsymbol{\pi}}_m)\}_{m=1}^M)$ is a dataset in which the inputs $\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_M \in \mathcal{X}$ and their prediction uncertainties $\tilde{\boldsymbol{\pi}}_1, \ldots, \tilde{\boldsymbol{\pi}}_M \in \Delta_{K-1}$ have been synthetically generated from original samples with hard labels. Since original classification datasets have only hard labels corresponding to the apexes of Δ_{K-1}, we obtain labels on Δ_{K-1} by synthetically generating $\tilde{\mathcal{D}}$. Note that in practice we do not need to pre-determine M and to generate the whole of $\tilde{\mathcal{D}}$ before Step 2, since we can generate as many samples in $\tilde{\mathcal{D}}$ as we need during the optimization in Step 2. The details of generating samples in $\tilde{\mathcal{D}}$ are explained in Section 4.3.1.

In the following subsection, we show the following properties of the proposed model: 1) Given a classifier trained with ordinary cross-entropy loss on $\mathcal{D}$, the parameter $\hat{\boldsymbol{\theta}}_\alpha$, which is estimated in Step 1, is immediately obtained. 2) The $\hat{\boldsymbol{\theta}}_\alpha$ is optimal regardless of $\boldsymbol{\theta}_\lambda$. 3) Step 2 preserves the classification accuracy of the classifier in Step 1. These properties indicate that we can calibrate a given pre-trained classifier without changing its classification accuracy, where $\boldsymbol{\theta}_\lambda$ is introduced as a new parameter for better calibration.

4.2 Why STS Realizes Accuracy-Preserving Calibration

In this subsection, we present the reason why the probabilistic model and the optimization process introduced in Section 4.1 serve as classification and accuracy-preserving calibration. To show that Step 1 corresponds to classification, we discuss Eq. (1) under Eqs. (4) and (5). Let $\boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{\theta}_\alpha) = (e^{g_1(\boldsymbol{x})}, \dots, e^{g_K(\boldsymbol{x})})^\top$, where $\boldsymbol{g} : \mathcal{X} \to \mathbb{R}^K$ is a DNN model parameterized by $\boldsymbol{\theta}_\alpha$. In this case, Eq. (1) becomes the softmax function with $\boldsymbol{g}(\boldsymbol{x})$ as logits, from the following theorem.

Theorem 1 (Predictive distribution)**.** *We assume that $p(y|\boldsymbol{\pi})$ and $p(\boldsymbol{\pi}|\boldsymbol{x})$ are formulated as Eqs. (4) and (5). Then, Eq. (1) is given as follows.*[2]

$$\begin{aligned} p(y = k|\boldsymbol{x}) &= \int p(y = k|\boldsymbol{\pi}) p(\boldsymbol{\pi}|\boldsymbol{x}) d\boldsymbol{\pi} \\ &= \frac{\alpha_k(\boldsymbol{x}, \boldsymbol{\theta}_\alpha)}{\sum_{i=1}^K \alpha_i(\boldsymbol{x}, \boldsymbol{\theta}_\alpha)} \quad \text{for } \forall k \in \mathcal{Y}. \end{aligned} \tag{6}$$

The proof is shown in Appendix A.1. Note that Eq. (6) is independent of $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$. From Theorem 1, we can show that Step 1 is equivalent to the standard DNN training with the cross-entropy loss as follows.

Corollary 1 (Classification via the Concrete distribution)**.** *Let $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ and $\boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{\theta}_\alpha) = (e^{g_1(\boldsymbol{x})}, \dots, e^{g_K(\boldsymbol{x})})^\top$, where $\boldsymbol{g} : \mathcal{X} \to \mathbb{R}^K$ is a function parameterized by $\boldsymbol{\theta}_\alpha$. Under Eqs. (4) and (5), optimizing $\boldsymbol{\theta}_\alpha$ on $\mathcal{D}$ using the maximum likelihood estimation of Eq. (1) is equivalent to optimizing $\boldsymbol{\theta}_\alpha$ on $\mathcal{D}$ using the minimization of the following criterion.*

$$\mathcal{L}(\boldsymbol{\theta}_\alpha) = -\frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K \mathbb{1}_{\{y_n = k\}} \ln \frac{e^{g_k(\boldsymbol{x}_n)}}{\sum_{i=1}^K e^{g_i(\boldsymbol{x}_n)}}. \tag{7}$$

Remark 2 (Interpretations of Corollary 1)**.** Corollary 1 shows that the weight parameters of a DNN model trained with the cross-entropy loss can be regarded as $\hat{\boldsymbol{\theta}}_\alpha$, which is estimated in Step 1. If the pre-trained DNN model is denoted by $\hat{\boldsymbol{g}} : \mathcal{X} \to \mathbb{R}^K$, we have $\boldsymbol{\alpha}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\alpha) = (e^{\hat{g}_1(\boldsymbol{x})}, \dots, e^{\hat{g}_K(\boldsymbol{x})})^\top$. Therefore, we can reuse the pre-trained DNN model for $\boldsymbol{\alpha}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\alpha)$ and skip Step 1. The proof is shown in Appendix A.2.

The maximality of the likelihood of Eq. (1) (i.e., the minimality of the cross-entropy loss) is guaranteed if $\boldsymbol{\theta}_\lambda$ shifts, because Eq. (6) is independent of $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$. This fact motivates us to fix $\boldsymbol{\alpha}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\alpha)$ in Step 2. As mentioned in Remark 1, we can estimate the prediction uncertainty by optimizing $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$ on $\tilde{\mathcal{D}}$, even when $\boldsymbol{\alpha}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\alpha)$ is fixed. Therefore, Step 2 realizes a calibration based on the prediction uncertainty.

[2]For confidence, another criterion is applied instead of Eq. (1). The confidence is explained in Section 4.4.

Preserved accuracy. If we consider predicting the class using Eq. (1) after Step 2, the classification accuracy of the pre-trained DNN model is preserved. According to Theorem 1, if we have $\hat{\boldsymbol{\theta}}_\alpha$, the predicted class is computed as

$$\arg\max_{k \in \mathcal{Y}} p(y = k|\boldsymbol{x}) = \arg\max_{k \in \mathcal{Y}} \frac{\alpha_k(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\alpha)}{\sum_{i=1}^K \alpha_i(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\alpha)}. \tag{8}$$

Equation (8) is equal to the class predicted by the pre-trained DNN model $\hat{\boldsymbol{g}}$ because $\boldsymbol{\alpha}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\alpha) = (e^{\hat{g}_1(\boldsymbol{x})}, \dots, e^{\hat{g}_K(\boldsymbol{x})})^\top$ as in Remark 2. Moreover, this prediction does not change when $\boldsymbol{\theta}_\lambda$ is updated in Step 2. As a result, the classification accuracy of the pre-trained DNN model is preserved.

Problems with the Dirichlet distribution. The probabilistic model with the Dirichlet distribution as in Eq. (2) also leads to an equation similar to Eq. (6) such as

$$\begin{aligned} p(y = k|\boldsymbol{x}) &= \int p(y = k|\boldsymbol{\pi}) p(\boldsymbol{\pi}|\boldsymbol{x}) d\boldsymbol{\pi} \\ &= \frac{\mu_k(\boldsymbol{x}, \boldsymbol{\theta}_\mu)}{\sum_{i=1}^K \mu_i(\boldsymbol{x}, \boldsymbol{\theta}_\mu)} \quad \text{for } \forall k \in \mathcal{Y}. \end{aligned} \tag{9}$$

If $\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\theta}_\mu) = (e^{g_1(\boldsymbol{x})}, \dots, e^{g_K(\boldsymbol{x})})^\top$, where $\boldsymbol{g}$ is parameterized by $\boldsymbol{\theta}_\mu$, Eq. (9) becomes the softmax function. Therefore, the maximum likelihood estimation of Eq. (1) is equivalent to minimizing the cross-entropy loss, as in STS. However, the Dirichlet distribution does not have a temperature parameter independent of $\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\theta}_\mu)$. Even if we introduce a temperature parameter in $\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\theta}_\mu)$, the maximality of the likelihood of Eq. (1) (i.e., the minimality of the cross-entropy loss) is not guaranteed after optimizing the temperature parameter solely using the maximum likelihood estimation of $\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\theta}_\mu))$.[3] Our STS solves this problem by replacing the Dirichlet distribution with the Concrete distribution and enables us to perform an accuracy-preserving calibration.

4.3 Training of the Temperature Parameter

This subsection describes the dataset and objective function of Step 2.

4.3.1 Generation of Samples on Probability Simplex by Multi-Mixup

To estimate $\boldsymbol{\theta}_\lambda$ by the maximum likelihood process, we require a dataset $\tilde{\mathcal{D}}$ in which inputs are labeled with vectors on the probability simplex that indicate the prediction uncertainty. As the model must distinguish inputs with large prediction uncertainties from those

[3]An example is presented in Appendix D.

with small prediction uncertainties, the dataset should include both input types. As in explained Section 3.2, previous calibration methods using Eq. (2) train ensemble models and label the inputs with the softmax outputs of the trained ensemble models, thus obtaining a dataset that satisfies the above properties (Malinin et al., 2020; Lindqvist et al., 2020; Ryabinin et al., 2021). Composing a dataset through this process requires training a few tens of DNN models initialized with different random weight parameters, which incurs a large training overhead. In addition, ensemble models that follow a pre-trained DNN model are difficult to obtain when the pre-trained DNN model is published online.

To solve these problems, we propose a method that synthetically generates training samples with labels on the probability simplex. Our Multi-Mixup method randomly selects one sample from each class $k \in \mathcal{Y}$ and linearly interpolates the inputs and one-hot labels of the selected samples. Let $\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(K)}$ be one-hot vectors indicating the classes. The k-th element of $\boldsymbol{y}^{(k)}$ is 1 and all other elements are 0. Now let $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(K)}$ be the inputs in $\mathcal{X}$ corresponding to $\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(K)}$. Multi-Mixup performs the following interpolation:

$$\tilde{\boldsymbol{x}} = \sum_{k=1}^{K} w_k \boldsymbol{x}^{(k)}, \qquad \tilde{\boldsymbol{\pi}} = \sum_{k=1}^{K} w_k \boldsymbol{y}^{(k)}, \tag{10}$$

where $\boldsymbol{w}$ is a random variable on Δ_{K-1}. The details of Multi-Mixup, including how to determine the value of $\boldsymbol{w}$, are explained in Appendix B.

Multi-Mixup can automatically generate inputs with small or large prediction uncertainties, along with labels indicating their prediction uncertainties. For example, we consider two weights, $\boldsymbol{w} = (0.01, 0.01, 0.98)^\top$ and $\boldsymbol{w} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})^\top$, under $K = 3$. The first weight generates an input with a small prediction uncertainty near the input $\boldsymbol{x}^{(3)}$, namely, $\tilde{\boldsymbol{x}} = 0.01\boldsymbol{x}^{(1)} + 0.01\boldsymbol{x}^{(2)} + 0.98\boldsymbol{x}^{(3)}$. The second weight generates an input with a large prediction uncertainty far from any of the three inputs $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}$, or $\boldsymbol{x}^{(3)}$, which is computed as $\tilde{\boldsymbol{x}} = \frac{1}{3}(\boldsymbol{x}^{(1)} + \boldsymbol{x}^{(2)} + \boldsymbol{x}^{(3)})$. These two inputs are labeled with $\tilde{\boldsymbol{\pi}} = (0.01, 0.01, 0.98)^\top$ and $\tilde{\boldsymbol{\pi}} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})^\top$, respectively. These labels indicate the prediction uncertainty of the two inputs.

Problems with existing data augmentations. Mixup (Zhang et al., 2018) generates samples only on the edges of Δ_{K-1}, which is inadequate for optimizing the parameter of a probabilistic model on Δ_{K-1}. Although AdaMixup (Guo et al., 2019) and ζ-Mixup (Abhishek et al., 2022) generate samples in the interior of Δ_{K-1}, the weights of the interpolations are constrained to restricted areas. This constraint is inappropriate for calibration because both inputs with large and small prediction uncertainty should be learned during calibration.

4.3.2 Loss Function for Training

Using the maximum likelihood estimation of Eq. (5), Step 2 of our method estimates $\boldsymbol{\theta}_\lambda$ on $\tilde{\mathcal{D}}$ generated by Multi-Mixup. To estimate the maximum likelihood on $\tilde{\mathcal{D}}$, we compute the gradient descent (Polak, 1997) of the following negative log-likelihood.

$$\begin{aligned}
\mathcal{L}_{\tilde{\mathcal{D}}}(\boldsymbol{\theta}_\lambda) = &-\frac{1}{M}\sum_{m=1}^{M}(K-1)\ln\lambda(\tilde{\boldsymbol{x}}_m, \boldsymbol{\theta}_\lambda) \\
&- \frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K}\ln\alpha_k(\tilde{\boldsymbol{x}}_m, \hat{\boldsymbol{\theta}}_\alpha) \\
&+ \frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K}(\lambda(\tilde{\boldsymbol{x}}_m, \boldsymbol{\theta}_\lambda)+1)\ln\tilde{\pi}_{m,k} \\
&+ \frac{1}{M}\sum_{m=1}^{M} K\ln\left(\sum_{i=1}^{K}\alpha_i(\tilde{\boldsymbol{x}}_m, \hat{\boldsymbol{\theta}}_\alpha)\tilde{\pi}_{m,i}^{-\lambda(\tilde{\boldsymbol{x}}_m, \boldsymbol{\theta}_\lambda)}\right),
\end{aligned} \tag{11}$$

where $\tilde{\mathcal{D}} = \{(\tilde{\boldsymbol{x}}_m, \tilde{\boldsymbol{\pi}}_m)\}_{m=1}^M$ and $\tilde{\pi}_{m,k}$ is the k-th entry of $\tilde{\boldsymbol{\pi}}_m$. Equation (11) is derived by taking the negative log of Eq. (3).

4.4 Confidence

In previous studies, the maximum of the predictive distribution $\max_{k\in\mathcal{Y}} p(y=k|\boldsymbol{x})$ is commonly used as confidence (Filho et al., 2021). However, in STS, this metric is computed as

$$\max_{k\in\mathcal{Y}} p(y=k|\boldsymbol{x}) = \max_{k\in\mathcal{Y}} \frac{\alpha_k(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\alpha)}{\sum_{i=1}^{K}\alpha_i(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\alpha)} \tag{12}$$

$$= \max_{k\in\mathcal{Y}} \frac{e^{\hat{g}_k(\boldsymbol{x})}}{\sum_{i=1}^{K} e^{\hat{g}_i(\boldsymbol{x})}}, \tag{13}$$

because $p(y=k|\boldsymbol{x})$ is derived as shown in Theorem 1. This means that if we use $\max_{k\in\mathcal{Y}} p(y=k|\boldsymbol{x})$ as confidence after Step 2, we cannot improve the confidence of the pre-trained DNN model $\hat{\boldsymbol{g}}$. Therefore, we propose to use another metric as confidence instead of $\max_{k\in\mathcal{Y}} p(y=k|\boldsymbol{x})$. Let $p(\boldsymbol{\pi}|\boldsymbol{x}) = \mathrm{Cn}(\boldsymbol{\pi}|\boldsymbol{\alpha}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\alpha), \lambda(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\lambda))$. The proposed confidence is defined as follows.[4]

$$\mathrm{confidence}(\boldsymbol{x}) \coloneqq \max_{k\in\mathcal{Y}} (\mathbb{E}[\boldsymbol{\pi}])_k \tag{14}$$

$$= \max_{k\in\mathcal{Y}} \left(\int \boldsymbol{\pi} p(\boldsymbol{\pi}|\boldsymbol{x}) d\boldsymbol{\pi}\right)_k, \tag{15}$$

[4]The confidence shown in Eq. (15) measures the sum of the aleatoric uncertainty and the epistemic uncertainty. Appendix E describes how to measure the two uncertainties separately after we obtain $\mathrm{Cn}(\boldsymbol{\pi}|\boldsymbol{\alpha}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\alpha), \lambda(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\lambda))$.

where $(\cdot)_k$ is the k-th entry of the vector in parentheses and $\mathbb{E}[\cdot]$ is the expectation of $p(\boldsymbol{\pi}|\boldsymbol{x})$. The motivation for using $\max_{k\in\mathcal{Y}}(\mathbb{E}[\boldsymbol{\pi}])_k$ as confidence is that in the categorical-Dirichlet model in Eq. (2), $\max_{k\in\mathcal{Y}}(\mathbb{E}[\boldsymbol{\pi}])_k$ becomes equivalent to $\max_{k\in\mathcal{Y}} p(y=k|\boldsymbol{x})$, the maximum of Eq. (9).

The expectation of the Concrete distribution, which cannot be computed analytically, is approximated as the sample mean of the values randomly sampled from $\mathrm{Cn}(\boldsymbol{\pi}|\boldsymbol{\alpha}(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\alpha),\lambda(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\lambda))$. According to Proposition 1(a) in Maddison et al. (2017), the Concrete distribution can be sampled using the standard Gumbel distribution (Gumbel, 1941). Therefore, the confidences can be computed as follows, where $j=1,\ldots,p$ and $k=1,\ldots,K$.

$$G_k^{(j)} \sim \mathrm{Gumbel}(0,1), \tag{16}$$

$$\hat{\pi}_k^{(j)}(\boldsymbol{x}) = \frac{\exp\left(\frac{\ln\alpha_k(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\alpha)+G_k^{(j)}}{\lambda(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\lambda)}\right)}{\sum_{i=1}^{K}\exp\left(\frac{\ln\alpha_i(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\alpha)+G_i^{(j)}}{\lambda(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\lambda)}\right)}, \tag{17}$$

$$\mathrm{confidence}(\boldsymbol{x}) \simeq \max_{k\in\mathcal{Y}}\frac{1}{p}\sum_{j=1}^{p}\hat{\pi}_k^{(j)}(\boldsymbol{x}). \tag{18}$$

In Eq. (16), $\mathrm{Gumbel}(0,1)$ denotes the standard Gumbel distribution. This expression states that pseudo-random numbers are generated $p\times K$ times from the standard Gumbel distribution. We set $p=30$ in the experiments of Section 5. The forward computations of $\boldsymbol{\alpha}(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\alpha)$ and $\lambda(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\lambda)$ are performed only once when computing the confidence of each input. Therefore, the computational overhead required to calculate confidence is negligible. If confidence is not required, the forward computation of the pre-trained DNN model is sufficient for predicting the class of $\boldsymbol{x}$, negating the need for sampling, as explained in Section 4.2. As discussed in Remark 1, the temperature parameter $\lambda(\boldsymbol{x},\boldsymbol{\theta}_\lambda)$ can represent the prediction uncertainty of $\boldsymbol{x}$, and this parameter is optimized by Multi-Mixup, which reflects the prediction uncertainty of the training samples, in Step 2. Therefore, Eq. (18) calculated with $\lambda(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\lambda)$ can approximate the classification accuracy more strictly than the confidence in previous studies. This fact is borne out by the experimental results in Section 5.

4.5 Architectures for Parameters in Probabilistic Model

As explained in Section 4.2, we can reuse a pre-trained DNN model for $\boldsymbol{\alpha}(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\alpha)$ and skip Step 1. We compute the location parameter such as

$$\boldsymbol{\alpha}(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\alpha) = (e^{\hat{g}_1(\boldsymbol{x})},\ldots,e^{\hat{g}_K(\boldsymbol{x})})^\top, \tag{19}$$

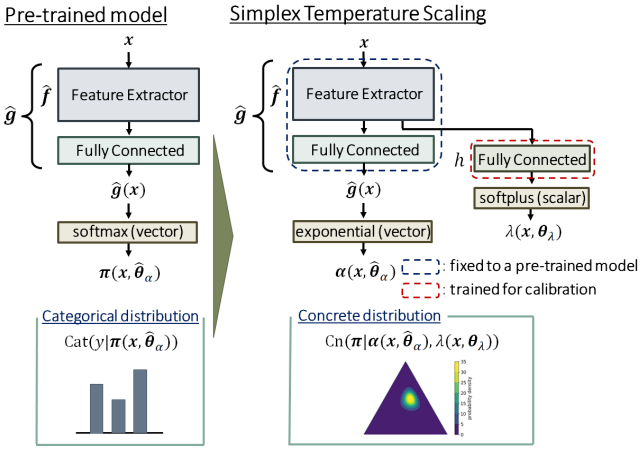


Figure 1: Overview of our proposed method, Simplex Temperature Scaling (STS). The Concrete distribution has two parameters, which are computed using a given pre-trained DNN model and an additional branch.

where $\hat{\boldsymbol{g}}:\mathcal{X}\to\mathbb{R}^K$ is a pre-trained DNN model. In addition, we introduce another DNN model for $\lambda(\boldsymbol{x},\boldsymbol{\theta}_\lambda)$ and train it in Step 2. Since many DNN models for classification include a feature extractor (Tan et al., 2018), we assume that the DNN model for $\lambda(\boldsymbol{x},\boldsymbol{\theta}_\lambda)$ shares its feature extractor with $\hat{\boldsymbol{g}}$. This sharing helps to reduce the computational complexity of calibration. Let $\hat{\boldsymbol{f}}:\mathcal{X}\to\mathcal{Z}$ be the feature extractor of $\hat{\boldsymbol{g}}$, where $\mathcal{Z}$ is a feature space. In other words, $\hat{\boldsymbol{f}}(\boldsymbol{x})$ denotes the outputs of a hidden layer in $\hat{\boldsymbol{g}}$. The temperature parameter is computed as

$$\lambda(\boldsymbol{x},\boldsymbol{\theta}_\lambda) = \mathrm{softplus}((h\circ\hat{\boldsymbol{f}})(\boldsymbol{x})), \tag{20}$$

where $\mathrm{softplus}(\cdot):\mathbb{R}\to(0,\infty)$ is the softplus function, which is defined as $\mathrm{softplus}(x)=\ln(1+e^x)$ (Glorot et al., 2011), and $h:\mathcal{Z}\to\mathbb{R}$ is a scalar-valued function computed by fully connected layers.

Although $\boldsymbol{\theta}_\lambda$ contains the weight parameters in $\hat{\boldsymbol{f}}$ and h, we optimize only the weight parameters of h, where $\hat{\boldsymbol{f}}$ is fixed, in Step 2. Figure 1 shows the pre-trained DNN model (left side) and the accuracy-preserving calibration by STS with the additional branch h (right side). Based on Corollary 1, we replace the softmax function of the pre-trained DNN model with the exponential function to obtain $\boldsymbol{\alpha}(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\alpha)$ and add a branch to a hidden layer of the pre-trained DNN model for $\lambda(\boldsymbol{x},\boldsymbol{\theta}_\lambda)$.

5 EXPERIMENTS

The superiority of STS was demonstrated in numerical comparison experiments of STS and existing accuracy-preserving calibration methods. For the numerical experiments, we selected two recently pro-

Table 1: Expected Calibration Errors (ECEs) and accuracies of the test dataset after calibration by Parameterized Temperature Scaling (PTS) [ECCV2022] (Tomani et al., 2022), Adaptive Temperature Scaling (AdaTS) [AAAI2023] (Joy et al., 2023), and Simplex Temperature Scaling (STS).

		ECE (%)				accuracy (%)
		pre-trained	PTS	AdaTS	STS (ours)	
FMNIST	LeNet5	3.52±0.15	**0.95±0.32**	**0.93±0.28**	**0.91±0.15**	91.2±0.17
	ResNet18	3.86±0.31	2.32±0.37	**1.15±0.16**	**1.15±0.28**	94.0±0.15
CIFAR10	VGG16-BN	4.40±0.14	2.48±0.24	1.90±0.54	**1.48±0.35**	93.7±0.19
	ResNet18	2.80±0.17	**1.28±0.18**	**1.21±0.38**	**1.10±0.24**	95.1±0.14
CIFAR100	ResNet50	9.04±0.36	6.71±0.56	**4.03±2.04**	**3.73±1.43**	78.7±0.72
	DenseNet121	7.23±0.59	5.71±0.81	4.22±1.08	**2.87±0.25**	79.6±0.50
STL10	VGG16-BN	15.23±0.56	3.84±0.60	4.41±1.75	**2.15±0.58**	79.4±0.38
	ResNet18	12.88±0.55	2.01±0.55	2.20±0.58	**1.48±0.29**	77.8±0.83

posed TS methods: Parameterized Temperature Scaling (PTS) (Tomani et al., 2022) and Adaptive Temperature Scaling (AdaTS) (Joy et al., 2023). The same pre-trained DNN model was used in each method and all methods were calibrated. Their calibration performances were then compared. The classification accuracy is shared among all methods. All calculations were performed on a NVIDIA A100 GPU.

5.1 Datasets

Experiments were performed on four open datasets for image classification, namely, FashionMNIST (FMNIST) (Xiao et al., 2017), CIFAR10, CIFAR100 (Krizhevsky and Hinton, 2009), and STL10 (Coates et al., 2011). The same datasets were employed in a previous calibration study (Thulasidasan et al., 2019). Each dataset was randomly split into a training dataset, a validation dataset, and a test dataset. The numbers of samples in the training, validation, and test datasets are given in Appendix C.1. The models were pre-trained on the training dataset prior to calibration, and the parameters for calibration were tuned on the validation dataset. While PTS and AdaTS were calibrated using the original samples with hard labels, STS was calibrated by applying Multi-Mixup to the original samples. The calibration performance was evaluated on the test dataset.

5.2 Setup for Training

We adopted LeNet5 (Lecun et al., 1998), ResNet18, ResNet50 (He et al., 2016), VGG16-BN (Simonyan and Zisserman, 2014), and DenseNet121 (Huang et al., 2017) architectures for the DNN models during pre-training. VGG16-BN defines the VGG16 (Simonyan and Zisserman, 2014) architecture with batch normalization. The pairings between the datasets and architectures are shown in Table 1. We applied different architectures for each dataset, depending on the image size and the difficulty of the tasks. During calibration, STS used the pre-trained DNN models for the location parameter, and PTS and AdaTS used them for logits. The architecture of the additional branch for the temperature parameter in STS is explained in Appendix C.2. The architecture for the temperature parameter in PTS was implemented as described in Tomani et al. (2022), and that in AdaTS was implemented as in the open source code (Joy, 2022).

All architectures on all datasets used the stochastic gradient descent (Bottou, 2010) optimizer for pre-training. The learning rate was reduced from 0.1 to 0 by cosine annealing (Loshchilov and Hutter, 2017). The momentum was 0.9, the weight decay was 0.0005, the batch size was 128, and the number of epochs was 200. The branch for the temperature parameter in STS was trained using the Adam optimizer (Kingma and Ba, 2014) with a fixed learning rate (0.001), a weight decay of 0.0005, and a batch size of 100. The number of epochs was 500. In each optimization step, 100 samples were generated by Multi-Mixup. The reason for applying the Adam optimizer is to reduce the impact of the learning rate. The temperature parameter in PTS was optimized as described in Tomani et al. (2022). For AdaTS, we used the hyperparameter values set in the open source code (Joy et al., 2023) to optimize the temperature parameter.

5.3 Evaluation

To evaluate the calibration performances of the methods, we adopted the Expected Calibration Error (ECE) (Guo et al., 2017), which divides samples into multiple bins of confidence levels and measures the dif-

ference between the confidence range in each bin and the accuracy of the samples in each bin. A lower ECE indicates a higher calibration performance.

In the experiments, we computed the accuracy and ECE using the test dataset. The number of bins for ECE was set to 10. In each method, we ran 5 trials with different random seeds and calculated the mean and standard deviation of the accuracies and ECEs over the 5 trials. The number of trials for the pre-training and the subsequent accuracy-preserving calibration are the same, and their random seeds are the same. The partitioning of the training, validation, and test datasets was fixed across the 5 trials.

5.4 Results

Table 1 shows the mean and standard deviation of the ECEs and accuracies in the methods.[5] Bold indicates the best results or results within one standard deviation of the best results. As shown in Table 1, STS consistently outperformed the other models, indicating that the performance of STS matches or surpasses the performances of PTS and AdaTS. This result, obtained by estimating the prediction uncertainty from the probabilistic model on the probability simplex, confirms the superiority of STS over PTS and AdaTS for calibration.

5.5 Accuracy-Preserving Calibration via Dirichlet Distribution

As we mentioned in Section 1, one possible approach to realize an accuracy-preserving calibration with a probabilistic model on the probability simplex is combining TS and the Dirichlet distribution. However, this method has problems as explained in Section 4.2. We experimentally confirmed that this method caused underconfidence after calibration. In the experiments using CIFAR10 and ResNet18, the mean of ECEs was 67.49 and the standard deviation was 3.84. In Appendix D, we explain the details of the accuracy-preserving calibration method with the Dirichlet distribution and the experiments to simulate this method.

5.6 Performance of Out-of-Distribution Detection

As explained in Section 3.2, the probabilistic model on the probability simplex can distinguish between aleatoric uncertainty and epistemic uncertainty. To confirm the benefit of this distinction, we performed the experiments of out-of-distribution (OOD) detection (Yang et al., 2021). The details of the experiments are explained in Appendix F. In OOD detection, estimating the epistemic uncertainty is important because OOD samples have high epistemic uncertainty for models trained with in-distribution samples. STS achieved higher OOD detection performance than PTS and AdaTS by computing the differential entropy (Cover and Thomas, 2006) of the estimated Concrete distribution, which is effective in estimating the epistemic uncertainty. The details of the differential entropy are discussed in Appendix E.

[5]Note that these values may differ from the results in Tomani et al. (2022) and Joy et al. (2023), because the setup of pre-training is different from these previous studies.

6 CONCLUSION

This paper proposed an accuracy-preserving calibration method called STS, which computes the confidence while considering the prediction uncertainty. STS estimates the prediction uncertainty by applying the Concrete distribution. The pre-trained DNN model requires no update in uncertainty estimation because it optimizes the location parameter of the Concrete distribution regardless of the temperature parameter. We also proposed a method called Multi-Mixup, which generates training samples labeled with vectors on the probability simplex. Multi-Mixup avoids the high training cost of previous methods that must train ensemble models. This advantage of Multi-Mixup extends to uncertainty estimation methods other than STS.

References

Abhishek, K., Brown, C. J., and Hamarneh, G. (2022). Multi-sample ζ-mixup: Richer, more realistic synthetic samples from a p-series interpolant. *arXiv preprint arXiv: 2204.03323*.

Balanya, S. A., Maroñas, J., and Ramos, D. (2022). Adaptive temperature scaling for robust calibration of deep neural networks. *arXiv preprint arXiv: 2208.00461*.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *International Conference on Computational Statistics*, pages 177–186. Physica-Verlag HD.

Boureau, Y., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *International Conference on Machine Learning*, page 111–118. Omnipress.

Bulatov, Y. (2011). notMNIST dataset. Google (Books/OCR), Tech. Rep. https://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html, Accessed on February 1, 2024.

Coates, A., Ng, A., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, volume 15, pages 215–223. PMLR.

Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory*. Wiley-Interscience.

Filho, T. S., Song, H., Perello-Nieto, M., Santos-Rodriguez, R., Kull, M., and Flach, P. (2021). Classifier calibration: A survey on how to assess and improve predicted class probabilities. *arXiv preprint arXiv: 2112.10327*.

Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., Shahzad, M., Yang, W., Bamler, R., and Zhu, X. X. (2021). A survey of uncertainty in deep neural networks. *arXiv preprint arXiv: 2107.03342*.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, volume 15, pages 315–323. PMLR.

Gumbel, E. J. (1941). The return period of flood flows. *The Annals of Mathematical Statistics*, 12(2):163–190.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International Conference on Machine Learning*, volume 70, pages 1321–1330. PMLR.

Guo, H., Mao, Y., and Zhang, R. (2019). Mixup as locally linear out-of-manifold regularization. *AAAI Conference on Artificial Intelligence*, 33(01):3714–3722.

Hüllermeier, E. and Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110:457–506.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Huang, G., Liu, Z., Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition*.

Inoue, H. (2018). Data augmentation by pairing samples for images classification. *arXiv preprint arXiv: 1801.02929*.

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Joy, T. (2022). Adaptive temperature scaling [Source code]. https://github.com/thwjoy/adats, Accessed on July 21, 2023.

Joy, T., Pinto, F., Lim, S., Torr, P. H., and Dokania, P. K. (2023). Sample-dependent adaptive temperature scaling for improved calibration. *AAAI Conference on Artificial Intelligence*, 37(12):14919–14926.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*.

Kristiadi, A., Hein, M., and Hennig, P. (2020). Being bayesian, even just a bit, fixes overconfidence in ReLU networks. In *International Conference on Machine Learning*, volume 119, pages 5436–5446. PMLR.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Kuleshov, V. and Deshpande, S. (2022). Calibrated and sharp uncertainties in deep learning via density estimation. In *International Conference on Machine Learning*, volume 162, pages 11683–11693. PMLR.

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.

Lindqvist, J., Olmin, A., Lindsten, F., and Svensson, L. (2020). A general framework for ensemble distribution distillation. In *International Workshop on Machine Learning for Signal Processing*, pages 1–6.

Loshchilov, I. and Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2017). The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

Malinin, A. and Gales, M. (2018). Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Malinin, A., Mlodozeniec, B., and Gales, M. (2020). Ensemble distribution distillation. In *International Conference on Learning Representations*.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein,

N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Polak, E. (1997). *Optimization : Algorithms and Consistent Approximations*. Springer-Verlag.

Powers, D. M. W. (2020). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv: 2010.16061*.

Ramalho, T. and Miranda, M. (2020). Density estimation in representation space to predict model uncertainty. In *Engineering Dependable and Secure Machine Learning Systems*, pages 84–96. Springer International Publishing.

Ryabinin, M., Malinin, A., and Gales, M. (2021). Scaling ensemble distribution distillation to many classes with proxy targets. In *Advances in Neural Information Processing Systems*, volume 34, pages 6023–6035. Curran Associates, Inc.

Sensoy, M., Kaplan, L., and Kandemir, M. (2018). Evidential deep learning to quantify classification uncertainty. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv: 1409.1556*.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C. (2018). A survey on deep transfer learning. In *International Conference on Artificial Neural Networks*, pages 270–279. Springer International Publishing.

Thulasidasan, S., Chennupati, G., Bilmes, J. A., Bhattacharya, T., and Michalak, S. (2019). On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Tomani, C., Cremers, D., and Buettner, F. (2022). Parameterized temperature scaling for boosting the expressive power in post-hoc uncertainty calibration. In *European Conference on Computer Vision*, pages 555–569. Springer Nature Switzerland.

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., and Bengio, Y. (2019). Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, volume 97, pages 6438–6447. PMLR.

Wenger, J., Kjellström, H., and Triebel, R. (2020). Non-parametric calibration for classification. In *International Conference on Artificial Intelligence and Statistics*, volume 108, pages 178–190. PMLR.

Wilson, A. G. and Izmailov, P. (2020). Bayesian deep learning and a probabilistic perspective of generalization. In *Advances in Neural Information Processing Systems*, volume 33, pages 4697–4708. Curran Associates, Inc.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv: 1708.07747*. The MIT License (MIT) Copyright © 2017 Zalando SE, https://tech.zalando.com.

Yang, J., Zhou, K., Li, Y., and Liu, Z. (2021). Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv: 2110.11334*.

Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., and Xiao, J. (2015). Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv: 1506.03365*.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Zhang, J., Kailkhura, B., and Han, T. Y. (2020). Mix-n-Match : Ensemble and compositional methods for uncertainty calibration in deep learning. In *International Conference on Machine Learning*, volume 119, pages 11117–11128. PMLR.

Zou, Y., Yu, Z., Liu, X., Kumar, B. V., and Wang, J. (2019). Confidence regularized self-training. In *IEEE/CVF International Conference on Computer Vision*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. / Yes. This description is provided in Section 4.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. / No. We do not discuss complexity. The complexities of training ensemble models and Multi-Mixup differ because they depend on the number of ensemble models and the training setup.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including

external libraries. / No. Our ability to release code to the public is limited by confidentialities imposed by our institution. To enable the reproduction of our results, we provide our experimental details in Section 5.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. / Yes. Please refer to Sections 4.1 and 4.2.
   (b) Complete proofs of all theoretical results. / Yes. Please refer to Appendix A.
   (c) Clear explanations of any assumptions. / Yes. Please refer to Sections 4.1 and 4.2.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). / No. Our ability to release code to the public is limited by confidentialities imposed by our institution. To enable the reproduction of our results, we provide our experimental details in Section 5.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). / Yes. The details are provided in Section 5.1, Section 5.2, Appendix B, and Appendix C.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). / Yes. Please see Sections 5.3 and 5.4.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). / Yes. This description is provided at the beginning of Section 5.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. / Yes. Please refer to Sections 5.1 and 5.2.
   (b) The license information of the assets, if applicable. / Yes. FasionMNIST (Xiao et al., 2017) specifies a license, which is given in the References.
   (c) New assets either in the supplemental material or as a URL, if applicable. / Not Applicable
   (d) Information about consent from data providers/curators. / Not Applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. / Not Applicable

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. / Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. / Not Applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. / Not Applicable

A MISSING PROOFS

This section describes the proofs missing from the main paper.

A.1 Proof of Theorem 1

By Eqs. (4) and (5), the following holds, where $\mathbb{E}[\cdot]$ denotes the expectation of a stochastic variable $\boldsymbol{\pi} \sim \mathrm{Cn}(\boldsymbol{\pi}|\boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{\theta}_\alpha), \lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda))$ and $\mathbb{P}(\cdot)$ returns the probability that the event in the parentheses occurs under the stochastic variable $\boldsymbol{\pi} \sim \mathrm{Cn}(\boldsymbol{\pi}|\boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{\theta}_\alpha), \lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda))$.

$$
\begin{aligned}
p(y = k|\boldsymbol{x}) &= \int p(y = k|\boldsymbol{\pi}) p(\boldsymbol{\pi}|\boldsymbol{x}) d\boldsymbol{\pi} && (21)\\
&= \int \mathbb{1}_{\{\arg\max_{i \in \mathcal{Y}} \pi_i = k\}} \mathrm{Cn}(\boldsymbol{\pi}|\boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{\theta}_\alpha), \lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)) d\boldsymbol{\pi} && (22)\\
&= \mathbb{E}\left[\mathbb{1}_{\{\arg\max_{i \in \mathcal{Y}} \pi_i = k\}}\right] && (23)\\
&= \mathbb{P}\left(\arg\max_{i \in \mathcal{Y}} \pi_i = k\right) && (24)\\
&= \frac{\alpha_k(\boldsymbol{x}, \boldsymbol{\theta}_\alpha)}{\sum_{i=1}^K \alpha_i(\boldsymbol{x}, \boldsymbol{\theta}_\alpha)} \quad \text{for } \forall k \in \mathcal{Y}. && (25)
\end{aligned}
$$

Equation (22) holds by Eqs. (4) and (5). Equation (24) holds by the definition of the expectation. Equation (25) holds under Proposition 1(b) in Maddison et al. (2017). □

A.2 Proof of Corollary 1

Let $\boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta}_\alpha)$ be defined as follows.

$$
\boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta}_\alpha) \coloneqq \left(\frac{\alpha_1(\boldsymbol{x}, \boldsymbol{\theta}_\alpha)}{\sum_{k=1}^K \alpha_k(\boldsymbol{x}, \boldsymbol{\theta}_\alpha)}, \ldots, \frac{\alpha_K(\boldsymbol{x}, \boldsymbol{\theta}_\alpha)}{\sum_{k=1}^K \alpha_k(\boldsymbol{x}, \boldsymbol{\theta}_\alpha)} \right)^\top. \tag{26}
$$

From Theorem 1, Eq. (1) is described as follows, under Eqs. (4) and (5).

$$
p(y = k|\boldsymbol{x}) = \frac{\alpha_k(\boldsymbol{x}, \boldsymbol{\theta}_\alpha)}{\sum_{i=1}^K \alpha_i(\boldsymbol{x}, \boldsymbol{\theta}_\alpha)} = \pi_k(\boldsymbol{x}, \boldsymbol{\theta}_\alpha) \quad \text{for } \forall k \in \mathcal{Y}. \tag{27}
$$

Therefore, optimizing $\boldsymbol{\theta}_\alpha$ on $\mathcal{D}$ using the maximum likelihood estimation of Eq. (1) is equivalent to optimizing $\boldsymbol{\theta}_\alpha$ on $\mathcal{D}$ using the maximum likelihood estimation of $\mathrm{Cat}(y|\boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta}_\alpha)) \cdots$(A).

Meanwhile, since $\boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{\theta}_\alpha) = (e^{g_1(\boldsymbol{x})}, \ldots, e^{g_K(\boldsymbol{x})})^\top$, Eq. (7) is also described as follows.

$$
\mathcal{L}(\boldsymbol{\theta}_\alpha) = -\frac{1}{N} \sum_{n=1}^N \ln \mathrm{Cat}(y_n|\boldsymbol{\pi}(\boldsymbol{x}_n, \boldsymbol{\theta}_\alpha)). \tag{28}
$$

Equation (28) is the negative log-likelihood of $\mathrm{Cat}(y|\boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta}_\alpha))$. Therefore, optimizing $\boldsymbol{\theta}_\alpha$ on $\mathcal{D}$ using the minimization of Eq. (7) is equivalent to optimizing $\boldsymbol{\theta}_\alpha$ on $\mathcal{D}$ using the maximum likelihood estimation of $\mathrm{Cat}(y|\boldsymbol{\pi}(\boldsymbol{x}, \boldsymbol{\theta}_\alpha)) \cdots$(B).

From (A) and (B), we can prove Corollary 1. □

Algorithm 1: Algorithm to synthetically generate a dataset in which the inputs are labeled with vectors on the probability simplex by Multi-Mixup.

Input: $\mathcal{B}^{(k)} (k \in \mathcal{Y})$: mini-batch including the pairs of inputs and one-hot labels in each class
$(|\mathcal{B}^{(1)}| = \cdots = |\mathcal{B}^{(K)}| = R,\ R \in \mathbb{N})$
$\beta \in (0, \infty)$: hyperparameter to generate the weight $\boldsymbol{w}$ by random sampling
$S \in \mathbb{N}$: the number of iterations to randomly shuffle the samples.
Output: $\tilde{\mathcal{B}}$: mini-batch generated by Multi-Mixup

1 **for** $s = 1, \ldots, S$ **do**
2 $\boldsymbol{w} \sim \mathrm{Dir}(\boldsymbol{\pi}|\beta \mathbf{1}_K)$
3 **for** $k = 1, \ldots, K$ **do**
4 Randomly shuffle the samples in $\mathcal{B}^{(k)}$, and get $\mathcal{B}_s^{(k)} = \{(\boldsymbol{x}_{s,r}^{(k)}, \boldsymbol{y}_{s,r}^{(k)})\}_{r=1}^{R}$
5 **end**
6 **for** $r = 1, \ldots, R$ **do**
7 $\tilde{\boldsymbol{x}}_{s,r} \leftarrow \sum_{k=1}^{K} w_k \boldsymbol{x}_{s,r}^{(k)}$
8 $\tilde{\boldsymbol{\pi}}_{s,r} \leftarrow \sum_{k=1}^{K} w_k \boldsymbol{y}_{s,r}^{(k)}$
9 **end**
10 **end**
11 **return** $\tilde{\mathcal{B}} = \{(\tilde{\boldsymbol{x}}_{s,r}, \tilde{\boldsymbol{\pi}}_{s,r})\}_{s=1,\ldots,S, r=1,\ldots,R}$

B ALGORITHM FOR MULTI-MIXUP

Algorithm 1 shows the details of Multi-Mixup in each optimization step. We transform samples with one-hot labels in classification datasets to samples with labels on the probability simplex that indicate the prediction uncertainty. At each optimization step, we randomly sample K mini-batches $\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(K)}$, each containing samples in each class, from the classification dataset. Random sampling is performed so that all mini-batches have the same number of samples, which is denoted by $R \in \mathbb{N}$.[6] In Multi-Mixup, we linearly interpolate the $R \times K$ samples among classes (Lines 6-9), which is repeated S times. For each $s \in \{1, \ldots, S\}$ and $r \in \{1, \ldots, R\}$, $\boldsymbol{y}_{s,r}^{(k)}$ denotes a one-hot vector whose k-th entry is 1 and whose other entries are 0. In each of the S iterations, we generate the value of the weight $\boldsymbol{w}$ by randomly sampling from the Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\pi}|\beta \mathbf{1}_K)$ (Line 2), where $\mathbf{1}_K$ is a K-dimensional vector whose all entries are 1 and β is a hyperparameter in $(0, \infty)$. In addition, we randomly shuffle the order of samples in each mini-batch (Lines 3-5). In the experiments explained in Section 5, we generated 100 samples at each step of the optimization by setting $R = 10$ and $S = 10$ in Alg. 1. Other settings are discussed in Appendix G.1.

Note that, in Line 8, $\boldsymbol{y}_{s,r}^{(i)} \neq \boldsymbol{y}_{s,r}^{(j)}$ for $i \neq j$. If we use a different policy than Alg. 1 and randomly select K samples without distinguishing the classes, samples corresponding to the surfaces and edges of Δ_{K-1} would be generated when there are class overlaps among the samples. In this case, if the number of classes K is large, the proportion of samples on the interior of Δ_{K-1} becomes extremely small, and there is a risk of obtaining a model that only returns a temperature parameter value close to 0 after training. Therefore, we formulate Multi-Mixup so that samples in different classes are interpolated.

The random sampling of $\boldsymbol{w}$ by $\mathrm{Dir}(\boldsymbol{\pi}|\beta \mathbf{1}_K)$ is based on the fact that Mixup (Zhang et al., 2018) determines the weight of the interpolation by sampling from the beta distribution $\mathrm{Beta}(\pi|\beta, \beta)$. Although we can also generate $\boldsymbol{w}$ by sampling from the Concrete distribution, a temperature parameter value that represents a uniform distribution on Δ_{K-1} varies with the number of dimensions, K, and we cannot describe this value in a closed form (Maddison et al., 2017). Conversely, the Dirichlet distribution can represent a uniform distribution on Δ_{K-1} regardless of K if we specify $\beta = 1$. In the experiments explained in Section 5, we varied the value of β from 0.2 to 2.0 in 0.1 increments and tried STS for each setting. Then, we calculated ECE for each setting with the validation dataset[7] and selected the setting that gave the smallest ECE. Since there were some settings where the loss shown in Eq. (11) diverged during optimization, such settings were automatically excluded. Finally, we evaluated the selected setting on the test dataset. The sensitivity of β to ECE is discussed in Appendix C.3.

[6] Even if we use a class-imbalanced dataset, we still sample equally across classes. In this case, we upsample for classes with small sample sizes. The evaluation of STS in a class-imbalanced situation is left as future work.

[7] The validation dataset was used for both training the temperature parameter and tuning the hyperparameter.

Table 2: Number of samples in each dataset.

	training	validation	test
FashionMNIST (Xiao et al., 2017)	60000	5000	5000
CIFAR10 (Krizhevsky and Hinton, 2009)	50000	5000	5000
CIFAR100 (Krizhevsky and Hinton, 2009)	50000	5000	5000
STL10 (Coates et al., 2011)	5000	4000	4000

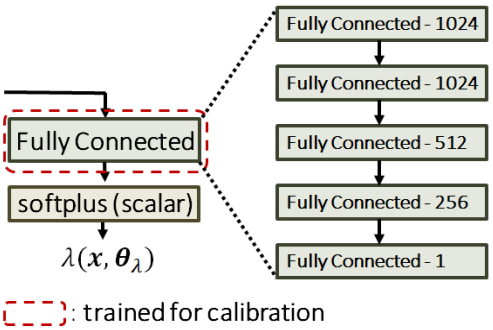


Figure 2: Architecture of the additional branch for the temperature parameter $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$ in the experiments. We used the same architecture regardless of the architecture of the pre-trained DNN model.

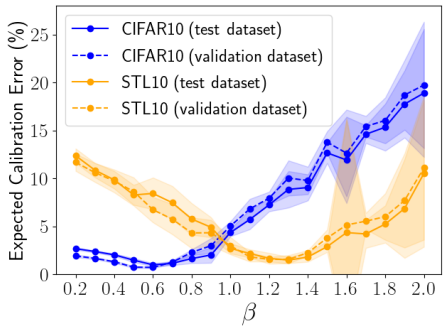


Figure 3: Transitions of Expected Calibration Errors (ECEs) when we varied the value of β from 0.2 to 2.0 in 0.1 increments. Each point indicates the median of 5 trials and the shading indicates the standard deviation.

C SUPPLEMENTS OF EXPERIMENTS IN THE MAIN PAPER

C.1 Dataset

Table 2 shows the number of samples in the training, validation, and test datasets used in the experiments explained in Section 5. The original files of FasionMNIST, CIFAR10, and CIFAR100 contain a training dataset and a test dataset, respectively. For the validation dataset, we randomly split the original test dataset into two datasets. The original file of STL10 contains a training dataset, a test dataset, and an unlabeled dataset. In this study, we did not use the unlabeled dataset and used only the training and test datasets. Similar to other datasets, we randomly split the original test dataset into two datasets for the validation dataset. The characteristic of STL10 is a small number of training samples, which causes overfitting to the training dataset. This characteristic is reflected in the test accuracy shown in Table 1.

C.2 Setup of the Models

As explained in Section 4.5, to compute the temperature parameter, we used an additional branch different from a pre-trained DNN model. Figure 2 shows the architecture of the branch to compute the temperature parameter. The same architecture was applied to all datasets and architectures of the pre-training. For the STL10 and VGG16-BN pairing only, we reduced the feature dimension from 512×3×3 to 512 by average pooling (Boureau et al., 2010). This process is due to the high resolution of STL10. The experiments with a different architecture are discussed in Appendix G.2.

C.3 Sensitivity of the Hyperparameter in Multi-Mixup

As explained in Appendix B, we selected the best value of the hyperparameter in Multi-Mixup, β, by the validation dataset. In this regard, we confirmed the difference in calibration performance among the tried values of β. Figure 3 shows the ECEs when we set 0.2-2.0 as the value of β. We chose the experiments with CIFAR10 and STL10 and used the models calibrated by STS with different values of β. The pre-training architecture was ResNet18. As can be seen in Fig. 3, the different datasets show different trends. While the best value of β is 0.6 on CIFAR10, that is 1.3 on STL10. However, the ECEs calculated on the validation dataset are consistent

Table 3: Time to calibrate confidence (training) and the time to compute confidence (inference).

	PTS	AdaTS	STS (ours)
training (sec.)	26142.47	269.88	5097.18
inference (sec.)	18.37	18.03	17.91

with those on the test dataset. This means that tuning β on the validation dataset results in high calibration performance on the test dataset, although this tuning is required on a per-task basis.

C.4 Time Required for Training and Inference

As part of a comparison with existing methods, we measured the time to calibrate confidence, which corresponds to training, and the time to compute confidence after calibration, which corresponds to inference, for PTS, AdaTS, and STS. We chose the experiments with the pair of CIFAR10 and ResNet18. The number of training epochs was set as described in Section 5.2. The time measurements were performed on a Xeon Platinum 8358 CPU. The number of workers in DataLoader (Paszke et al., 2019) and that of cores were unified to 8. We used a NVIDIA A100 GPU as described in the main paper. Table 3 shows the results of the time measurements. The time required for training varies greatly from method to method. This variation is caused by the difference in the objective function used to optimize the temperature parameter and the difference in the number of epochs due to the objective function. Although STS is shorter than PTS, that is longer than AdaTS. Therefore, we should devise a method to shorten the training time in future work. In contrast, there is little difference in the time required for inference among the methods. This implies that the sampling from the Gumbel distribution required to compute our confidence in STS does not hurt the running time compared to the existing methods.

D INAPPROPRIATENESS OF DIRICHLET DISTRIBUTION FOR ACCURACY-PRESERVING CALIBRATION

D.1 Calibration with Dirichlet Distribution

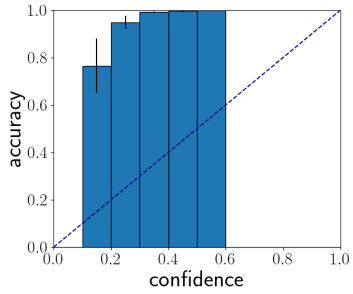
In this section, we explain why the probabilistic model as in Eq. (2) is inappropriate for accuracy-preserving calibration in order to reinforce the validity of adopting the Concrete distribution. We consider calibration using Eq. (2) as in previous studies (Malinin and Gales, 2018; Malinin et al., 2020; Lindqvist et al., 2020; Ryabinin et al., 2021). Moreover, we add a temperature parameter into the parameter of the Dirichlet distribution as

$$\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\theta}_\mu) = \left(\exp\left(\frac{\hat{g}_1(\boldsymbol{x})}{t(\boldsymbol{x}, \boldsymbol{\theta}_\mu)} \right), \cdots, \exp\left(\frac{\hat{g}_K(\boldsymbol{x})}{t(\boldsymbol{x}, \boldsymbol{\theta}_\mu)} \right) \right)^\top, \tag{29}$$

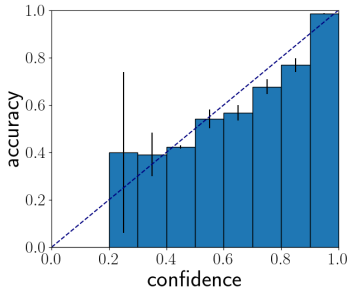
where $\hat{\boldsymbol{g}} : \mathcal{X} \to \mathbb{R}^K$ is a pre-trained DNN model and $t(\cdot, \boldsymbol{\theta}_\mu) : \mathcal{X} \to (0, \infty)$ is a scalar-valued function parameterized by a real vector $\boldsymbol{\theta}_\mu$. We determined the position of $t(\boldsymbol{x}, \boldsymbol{\theta}_\mu)$ in Eq. (29) based on the temperature annealing discussed in Malinin et al. (2020). In this case, the negative log-likelihood of $\mathrm{Dir}(\boldsymbol{\pi} | \boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\theta}_\mu))$ is formulated as

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}_\mu) = -\frac{1}{M} \sum_{m=1}^{M} \ln \Gamma \left(\sum_{k=1}^{K} \exp\left(\frac{\hat{g}_k(\tilde{\boldsymbol{x}}_m)}{t(\tilde{\boldsymbol{x}}_m, \boldsymbol{\theta}_\mu)} \right) \right) + \frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} \ln \Gamma \left(\exp\left(\frac{\hat{g}_k(\tilde{\boldsymbol{x}}_m)}{t(\tilde{\boldsymbol{x}}_m, \boldsymbol{\theta}_\mu)} \right) \right) \\ - \frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} \left(\exp\left(\frac{\hat{g}_k(\tilde{\boldsymbol{x}}_m)}{t(\tilde{\boldsymbol{x}}_m, \boldsymbol{\theta}_\mu)} \right) - 1 \right) \ln \tilde{\pi}_{m,k}, \end{aligned} \tag{30}$$

where $\Gamma(\cdot)$ is the gamma function and $\{(\tilde{\boldsymbol{x}}_m, \tilde{\boldsymbol{\pi}}_m)\}_{m=1}^M$ denotes the dataset generated by Multi-Mixup. For an accuracy-preserving calibration, the temperature parameter is computed as $t(\boldsymbol{x}, \boldsymbol{\theta}_\mu) = \mathrm{softplus}((h \circ \hat{\boldsymbol{f}})(\boldsymbol{x}))$, where $h : \mathcal{Z} \to \mathbb{R}$ is a scalar-valued function computed by fully connected layers. The weight parameters of h are optimized by minimizing Eq. (30), where $\hat{\boldsymbol{g}}$ and $\hat{\boldsymbol{f}}$ are fixed.



(a) Dirichlet distribution.



(b) Concrete distribution.

Figure 4: Relation between the confidence and classification accuracy when we use the probabilistic model as in Eq. (2) (left side) and the probabilistic model as in Eqs. (4) and (5) (right side). The interval between 0 and 1 is divided into 10 bins, and the height of the bars indicates the classification accuracy of the samples whose confidence is within each bin. The bins without a bar imply that there is no sample in the test dataset that has a confidence level within the corresponding bins. We plotted the mean of 5 trials with different random seeds, and the standard deviation is represented by error bars. The diagonal dashed line represents the ideal case where confidence and classification accuracy are in perfect agreement, and the closer the height of the bars is to the dashed line, the more accurate the calibration.

Let $t(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\mu)$ be the temperature parameter estimated by Eq. (30). We compute the confidence as follows.

$$\text{confidence}(\boldsymbol{x}) = \max_{k \in \mathcal{Y}} (\mathbb{E}[\boldsymbol{\pi}])_k = \max_{k \in \mathcal{Y}} \frac{\exp\left(\frac{\hat{g}_k(\boldsymbol{x})}{t(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\mu)}\right)}{\sum_{i=1}^{K} \exp\left(\frac{\hat{g}_i(\boldsymbol{x})}{t(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\mu)}\right)}, \tag{31}$$

where $\mathbb{E}[\cdot]$ denotes the expectation of $\mathrm{Dir}(\boldsymbol{\pi} | \boldsymbol{\mu}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\mu))$. This previous study takes advantage of the fact that the Dirichlet distribution is a conjugate prior, and Eq. (31) is equivalent to the maximum of Eq. (1).

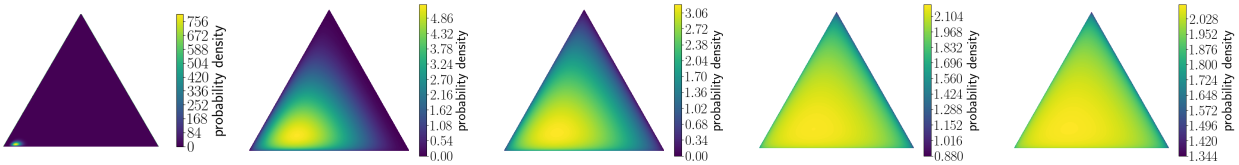
Using the Dirichlet distribution, the maximum likelihood estimation of Eq. (1) is equivalent to the training with the cross-entropy loss similar to STS. Therefore, if we denote $\boldsymbol{\mu}_{\hat{\boldsymbol{g}}}(\boldsymbol{x}, \boldsymbol{\theta}_\mu) = (e^{\hat{g}_1(\boldsymbol{x})}, \ldots, e^{\hat{g}_K(\boldsymbol{x})})^\top$, $\boldsymbol{\mu}_{\hat{\boldsymbol{g}}}(\boldsymbol{x}, \boldsymbol{\theta}_\mu)$ is optimal in terms of the likelihood of Eq. (1). However, the optimality of this parameter is not guaranteed after training $t(\cdot, \boldsymbol{\theta}_\mu)$ solely using Eq. (30). In other words, training $t(\cdot, \boldsymbol{\theta}_\mu)$ by Eq. (30) may make $\boldsymbol{\mu}_{\hat{\boldsymbol{g}}}(\boldsymbol{x}, \boldsymbol{\theta}_\mu)$ less optimal in terms of the cross-entropy loss. This problem results in poor calibration performance. We confirmed that the accuracy-preserving calibration with Eq. (30) leads to underconfidence by numerical experiments. The experiments are explained in the following section.

D.2 Experiments to Confirm Inappropriateness

We used CIFAR10 (Krizhevsky and Hinton, 2009) as a dataset and used ResNet18 (He et al., 2016) as a model architecture. The pre-trained DNN model $\hat{\boldsymbol{g}}$ was the same as in Section 5. The architecture for the temperature parameter $t(\cdot, \boldsymbol{\theta}_\mu)$ and the setup for training $t(\cdot, \boldsymbol{\theta}_\mu)$ and tuning the hyperparameter β were the same as in STS, except for the learning rate. The learning rate was set to 10^{-6} because the loss function (30) diverged when set to the same value as in STS. After training $t(\cdot, \boldsymbol{\theta}_\mu)$, we computed ECE with the test dataset.

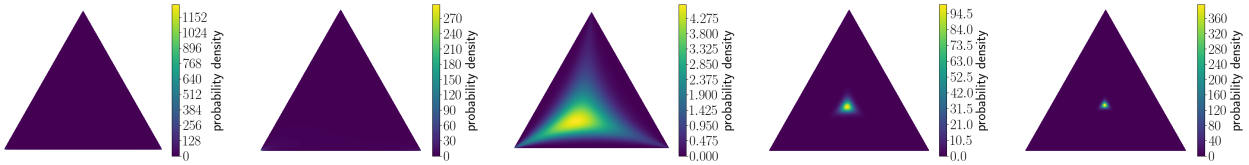
As a result, the mean of ECE was 67.49 and the standard deviation was 3.84, which is worse than the pre-trained DNN model. From this result, we can experimentally confirm that the Dirichlet distribution is inappropriate for accuracy-preserving calibration even if we add a temperature parameter to the Dirichlet distribution. Figure 4a implies the cause of the poorer ECE. We have plotted the relation between the confidence computed by Eq. (31) and the classification accuracy. For reference, we also present the same type of figure for STS, in Fig. 4b. From Fig. 4a, the accuracy-preserving calibration using the Dirichlet distribution results in confidence levels of only 0.6 or less although the classification accuracy of the pre-trained DNN model is about 95% as shown in Table 1. In other words, extreme underconfidence is observed.

As we mentioned in the main paper, the failure of the accuracy-preserving calibration when we used the Dirichlet distribution is caused by the fact that the maximality of the likelihood of Eq. (1) (i.e., the minimality of the



(a) $t(\boldsymbol{x}, \boldsymbol{\theta}_\mu) = 0.2$ (b) $t(\boldsymbol{x}, \boldsymbol{\theta}_\mu) = 1$ (c) $t(\boldsymbol{x}, \boldsymbol{\theta}_\mu) = 2$ (d) $t(\boldsymbol{x}, \boldsymbol{\theta}_\mu) = 10$ (e) $t(\boldsymbol{x}, \boldsymbol{\theta}_\mu) = 20$

Figure 5: Variation of the Dirichlet distribution when we shifted only $t(\boldsymbol{x}, \boldsymbol{\theta}_\mu)$ with fixed $\hat{\boldsymbol{g}}(\boldsymbol{x})$. In this figure, we did not use any input $\boldsymbol{x}$ and give pseudo values for $t(\boldsymbol{x}, \boldsymbol{\theta}_\mu)$ and $\hat{\boldsymbol{g}}(\boldsymbol{x})$. The value of $\hat{\boldsymbol{g}}(\boldsymbol{x})$ was fixed to $(1.0, 0.5, 0.25)^\top$. The value of $t(\boldsymbol{x}, \boldsymbol{\theta}_\mu)$ was shifted from 0.2 to 20.



(a) $\lambda = 0.2$ (b) $\lambda = 1$ (c) $\lambda = 2$ (d) $\lambda = 10$ (e) $\lambda = 20$

Figure 6: Variation of the Concrete distribution when we shifted only the temperature parameter λ with the fixed location parameter $\boldsymbol{\alpha}$. The value of $\boldsymbol{\alpha}$ was fixed to $(e^{1.0}, e^{0.5}, e^{0.25})^\top$. The value of λ was shifted from 0.2 to 20. Due to the high probability density, bright areas are almost non-existent in Figs. 6a and 6b. In these figures, the probability density is concentrated at the lower left apex.

cross-entropy loss) does not hold when only $t(\cdot, \boldsymbol{\theta}_\mu)$ is trained using Eq. (30). Moreover, even if only $t(\cdot, \boldsymbol{\theta}_\mu)$ increases with fixed $\hat{\boldsymbol{g}}$, the Dirichlet distribution cannot represent the case where the probability density of the center of Δ_{K-1} is high, depending on the value of $\hat{\boldsymbol{g}}(\boldsymbol{x})$, as shown in Fig. 5. Nevertheless, the samples corresponding to the center of Δ_{K-1} were given to the probabilistic model, causing $t(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\mu)$ to be larger than expected, resulting in underconfidence. On the contrary, the Concrete distribution can represent the case where the probability density of the center of Δ_{K-1} is high by shifting the temperature parameter λ alone, as shown in Fig. 6. Therefore, STS can calibrate the confidence accurately.

E ESTIMATION OF ALEATORIC AND EPISTEMIC UNCERTAINTIES

As explained in Section 3.2, the merit of the probabilistic model on Δ_{K-1} is that it can measure aleatoric uncertainty and epistemic uncertainty separately. A high probability density at the center of Δ_{K-1} corresponds to aleatoric uncertainty, and a large variance over Δ_{K-1} corresponds to epistemic uncertainty (Malinin et al., 2020). Let $p(\boldsymbol{\pi}|\boldsymbol{x})$ be a probabilistic model on Δ_{K-1} with $\boldsymbol{x}$-dependent parameters and $\mathbb{E}[\cdot]$ be the expectation of $p(\boldsymbol{\pi}|\boldsymbol{x})$. The aleatoric uncertainty is measured by the following expectation of entropy.

$$\mathbb{E}\left[-\sum_{k=1}^{K} \pi_k \ln \pi_k\right] = -\int \sum_{k=1}^{K} \pi_k \ln \pi_k \; p(\boldsymbol{\pi}|\boldsymbol{x}) d\boldsymbol{\pi}. \tag{32}$$

When $p(\boldsymbol{\pi}|\boldsymbol{x})$ has a high probability density at the center of Δ_{K-1}, Eq. (32) has a large value. In this case, the input $\boldsymbol{x}$ is judged to have high aleatoric uncertainty. The epistemic uncertainty is measured by the following differential entropy (Cover and Thomas, 2006).

$$\mathbb{E}[-\ln p(\boldsymbol{\pi}|\boldsymbol{x})] = -\int p(\boldsymbol{\pi}|\boldsymbol{x}) \ln p(\boldsymbol{\pi}|\boldsymbol{x}) d\boldsymbol{\pi}. \tag{33}$$

When $p(\boldsymbol{\pi}|\boldsymbol{x})$ has a large variance over Δ_{K-1}, Eq. (33) has a large value. In this case, the input $\boldsymbol{x}$ is judged to have high epistemic uncertainty.

In the case of the proposed probabilistic model shown in Eqs. (4) and (5), all we need to do is set $p(\boldsymbol{\pi}|\boldsymbol{x})$ to the estimated Concrete distribution $\mathrm{Cn}(\boldsymbol{\pi}|\boldsymbol{\alpha}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\alpha), \lambda(\boldsymbol{x}, \hat{\boldsymbol{\theta}}_\lambda))$, in the above metrics. As in Section 4.4, the

Table 4: Pairs of datasets for in-distribution and out-of-distribution (OOD).

	in-distribution	out-of-distribution
Set 1	FashionMNIST (Xiao et al., 2017)	notMNIST (Bulatov, 2011)
Set 2	CIFAR10 (Krizhevsky and Hinton, 2009)	SVHN (Netzer et al., 2011)
Set 3	CIFAR10 (Krizhevsky and Hinton, 2009)	LSUN (Yu et al., 2015)

Table 5: Area under the curve of precision-recall (AUPR) and that of receiver operating characteristic (AUROC) computed by the test datasets for in-distribution and out-of-distribution (OOD).

		pre-trained	PTS	AdaTS	STS (ours)	
					confidence	differential entropy
Set 1	AUPR (%)	97.80±0.48	97.97±0.48	96.80±0.94	98.02±0.53	**98.63±0.46**
	AUROC (%)	94.22±1.28	94.30±1.38	89.70± 3.30	**94.40±1.56**	**95.79±1.43**
Set 2	AUPR (%)	**98.74±0.21**	**98.84±0.20**	**98.89±0.20**	**98.81±0.20**	**98.93±0.34**
	AUROC (%)	**95.44±0.51**	**95.70±0.50**	**95.74±0.68**	**95.51±0.53**	**95.68±1.70**
Set 3	AUPR (%)	95.04±0.15	95.39±0.14	94.94±0.22	95.28±0.17	**96.58±0.21**
	AUROC (%)	92.64±0.17	92.93±0.18	92.22±0.25	92.59±0.23	**94.11±0.37**

expectation of the Concrete distribution is approximated as the sample mean of the values sampled by the softmax of the standard Gumbel distribution as follows.

$$\mathbb{E}\left[-\sum_{k=1}^{K}\pi_k \ln \pi_k\right] \simeq -\frac{1}{p}\sum_{j=1}^{p}\sum_{k=1}^{K}\hat{\pi}_k^{(j)}(\boldsymbol{x}) \ln \hat{\pi}_k^{(j)}(\boldsymbol{x}), \tag{34}$$

$$\mathbb{E}[-\ln p(\boldsymbol{\pi}|\boldsymbol{x})] \simeq -\frac{1}{p}\sum_{j=1}^{p} \ln \mathrm{Cn}(\boldsymbol{\pi}|\boldsymbol{\alpha}(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\alpha), \lambda(\boldsymbol{x},\hat{\boldsymbol{\theta}}_\lambda))\Big|_{\boldsymbol{\pi}=(\hat{\pi}_1^{(j)}(\boldsymbol{x}),\ldots,\hat{\pi}_K^{(j)}(\boldsymbol{x}))^\top}, \tag{35}$$

where $\hat{\pi}_k^{(j)}(\boldsymbol{x})$ is computed as Eq. (17).

F EFFECTIVENESS ON OUT-OF-DISTRIBUTION DETECTION

In the main paper, Table 1 shows that STS estimates classification accuracy more precisely than the previous methods. In addition, by computing the differential entropy shown in Eq. (35), STS can achieve higher performance in OOD detection than previous methods, because OOD detection is a task of estimating the epistemic uncertainty that OOD samples have (Gawlikowski et al., 2021). To show the validity of this claim, we confirmed the OOD detection performance of the pre-trained model and the models calibrated by PTS, AdaTS, and STS.

Table 4 shows the pairs of datasets for in-distribution and OOD. As shown in Table 4, we tried three pairs in our experiments, which are named Set 1, Set 2, and Set 3. Note that the datasets for OOD, notMNIST (Bulatov, 2011), SVHN (Netzer et al., 2011), and LSUN (Yu et al., 2015), were not used in either pre-training, calibration, or tuning of hyperparameters, and these datasets were used for performance evaluation purposes only. The number of samples in the test datasets of notMNIST, SVHN, and LSUN is 18724, 26032, and 10000, respectively. We adopted ResNet18 architecture for the DNN models during pre-training. The training setup was the same as in Section 5 and Appendix C. In other words, we reused the pre-trained model and the calibrated models obtained in the experiments described in the main paper, picking up CIFAR10 and FMNIST. In STS, we also reused the best value of β selected as explained in Appendix B. Therefore, we do not need to train another model for OOD detection. The OOD detection performance was evaluated by the area under the curve of precision-recall (AUPR) and that of receiver operating characteristic (AUROC) (Powers, 2020). The confidence of the pre-trained model and that of the models calibrated by PTS, AdaTS, and STS were used as OOD detectors. Moreover, differential entropy was also used as an OOD detector in STS. For differential entropy, the same value of β was adopted as in the case of confidence.

Table 6: Expected Calibration Errors (ECEs) of the test dataset when we performed calibration with different values of R and S. The results of the methods except STS are the same as in Table 1.

	pre-trained	PTS	AdaTS	STS (ours)		
				$R = 10, S = 10$	$R = 20, S = 5$	$R = 100, S = 1$
FMNIST	3.86±0.31	2.32±0.37	**1.15±0.16**	**1.15±0.28**	**1.01±0.16**	**1.04±0.26**
CIFAR10	2.80±0.17	**1.28±0.18**	**1.21±0.38**	**1.10±0.24**	**1.06±0.34**	**1.08±0.15**

Table 5 shows the mean and standard deviation of AUPR and AUROC in the methods. As in Table 1, bold indicates the best results or results within one standard deviation of the best results. Although there are no significant differences in Set 2 due to the simplicity of the task, STS shows higher performance than PTS and AdaTS in Set 1 and Set 3. In particular, differential entropy achieved the highest performance, confirming that the differential entropy of a distribution on the probability simplex is effective in OOD detection where a detector that estimates epistemic uncertainty is required. These results also follow up on existing studies claiming the validity of differential entropy on OOD detection (Malinin and Gales, 2018; Malinin et al., 2020).

G ADDITIONAL EXPERIMENTS WITH DIFFERENT SETUPS

This section describes calibration experiments with different setups than those described in the previous sections.

G.1 Sensitivity of Sampling for Multi-Mixup

As explained in Appendix B, in the previous sections we set $R = 10$, which is the number of samples in mini-batches $\mathcal{B}^{(k)}$ for each class, and $S = 10$, which is the number of times the interpolation is repeated. Since R and S are adjustable, we performed additional experiments to investigate the variation in calibration performance as these numbers are adjusted. We used CIFAR10 and FMNIST for the datasets and ResNet18 for the architecture in pre-training. For R and S, we tried two setups. One is $R = 20, S = 5$ and the other is $R = 100, S = 1$. All setups except R, S, and the number of epochs were not changed from the previous sections. The number of iterations in the optimization was adapted to the experiments in the main paper. Therefore, as we increased R, the number of epochs, which is the number of times each original pre-Multi-Mixup sample is accessed, was increased.

Table 6 shows the mean and standard deviation of the ECEs in the methods. There are no significant differences among the different R and S settings and the choices of these numbers have little effect on calibration. However, if we focus on the mean, the ECEs become smaller as R is increased. These results suggest the possibility of achieving smaller ECEs.

G.2 Sensitivity of the Architecture for the Temperature Parameter

In the previous sections, we used only the architecture shown in Fig. 2 for the temperature parameter $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$. Therefore, this section presents experiments using a different architecture to investigate the sensitivity of the architecture to $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$. As in Appendix G.1, we used CIFAR10 and FMNIST for the datasets and ResNet18 for the architecture in pre-training. Figure 7 shows a new architecture introduced for additional experiments. The new architecture has fewer layers and narrower widths than the architecture in Fig. 2. All setups except architectures were not changed from the previous sections. We adopted the $R = 10, S = 10$ setting.

Table 7 shows the mean and standard deviation of the ECEs in the methods. There are no significant differences among the different architectures and the choices of the architectures for $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$ have little effect on calibration. However, if we focus on the mean, the ECEs get larger as the architecture gets smaller. This suggests that there is a risk that the calibration performance will deteriorate as the architecture gets smaller, and we should make sure that the size of the architecture is sufficient for $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$ when we perform the calibration using STS.

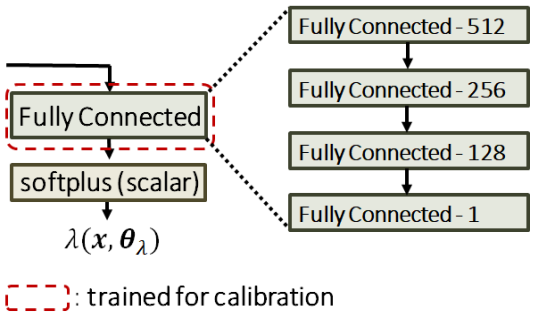


Figure 7: Architecture of the additional branch for the temperature parameter $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$, introduced for additional experiments described in Appendix G.2.

Table 7: Expected Calibration Errors (ECEs) of the test dataset when we performed calibration with different architectures for the temperature parameter $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$. The results of the methods except STS are the same as in Table 1. The columns indicated by "Figure 2" and "Figure 7" show the results with the different architectures.

	pre-trained	PTS	AdaTS	STS (ours)	
				Figure 2	Figure 7
FMNIST	3.86±0.31	2.32±0.37	**1.15±0.16**	**1.15±0.28**	**1.30±0.18**
CIFAR10	2.80±0.17	**1.28±0.18**	**1.21±0.38**	**1.10±0.24**	**1.11±0.16**