Journal of Machine Learning Research 26 (2025) 1-46 Submitted 5/24; Revised 6/25; Published 7/25

Extending Temperature Scaling with Homogenizing Maps

Christopher Qian CQ7@ILLINOIS.EDU
Department of Statistics
University of Illinois Urbana-Champaign
Champaign, IL 61820, USA

Feng Liang LIANGF@ILLINOIS.EDU
Department of Statistics
University of Illinois Urbana-Champaign
Champaign, IL 61820, USA

Jason Adams JRADAMS@SANDIA.GOV
Sandia National Laboratories
Albuquerque, NM 87123, USA

Editor: Chris Oates

Abstract

As machine learning models continue to grow more complex, poor calibration significantly limits the reliability of their predictions. Temperature scaling learns a single temperature parameter to scale the output logits, and despite its simplicity, remains one of the most effective post-hoc recalibration methods. We identify one of temperature scaling's defining attributes, that it increases the uncertainty of the predictions in a manner that we term homogenization, and propose to learn the optimal recalibration mapping from a larger class of functions that satisfies this property. We demonstrate the advantage of our method over temperature scaling in both calibration and out-of-distribution detection. Additionally, we extend our methodology and experimental evaluation to recalibration in the Bayesian setting.

Keywords: uncertainty quantification, recalibration, out-of-distribution, temperature scaling, entropy

1. Introduction

In supervised learning, probabilistic models are essential to improving the reliability of the predictions. In the context of multiclass classification, a model's *confidence*, the probability of the predicted class, can be used to determine whether or not to reject giving a prediction (Chow, 1957; Hüllermeier and Waegeman, 2021). In addition, the predicted probabilities can also be used to determine whether or not a test observation is *out-of-distribution* (OOD) (Hendrycks and Gimpel, 2017; Możejko et al., 2018). However, many modern machine learning models are overconfident, assigning high probabilities to blatantly incorrect predictions (Guo et al., 2017). This severely limits the usefulness of the models' probabilistic outputs for high-stakes settings, such as healthcare (Seoni et al., 2023). Such model overconfidence has led to the development of post-hoc *recalibration* methods, in which a recalibration mapping is learned from validation data to apply to the pre-trained model. Inspired by Platt scaling (Platt, 1999), which learns a logistic regression model to obtain probabilities from a sup-

port vector machine, *temperature scaling* (Guo et al., 2017) simply learns one temperature parameter to scale the logit predictions of a neural network.

Recent Recalibration Methods. Many other recalibration methods have been proposed to improve upon temperature scaling. These methods tend to be significantly more complex, often involving learning another neural network on the output logits (Rahimi et al., 2020). Joy et al. (2023), Balanya et al. (2024), and Ding et al. (2021) consider performing *adaptive* temperature scaling, where a different temperature is used for each input, which necessitates learning significantly more parameters based on the input observation/logit vector. Meanwhile, Kull et al. (2019) and Frenkel and Goldberger (2021) consider improving the class-wise performance of the base model, but this does not necessarily translate to better overall performance. In general, these methods focus on traditional calibration metrics such as expected calibration error; there has been less exploration of the effects of recalibration on *out-of-distribution* (OOD) detection performance.

We develop a recalibration method that improves the calibration performance of temperature scaling while retaining the OOD detection performance. To do so, we identify a process called *homogenization* that describes how the uncertainty in the prediction increases after applying temperature scaling, and we show that this process implies that the uncertainty also increases in terms of entropy and variation ratio. In our approach, we propose to learn a mapping from a softplus family of functions that that increase the uncertainty in this manner by constraining the slope of the candidate mappings. We experimentally demonstrate that using the more flexible softplus functions improves the in-distribution calibration performance, while enforcing homogenization maintains the OOD detection performance.

Some recent work that also discuss the effect of recalibration on OOD detection performance include Esaki et al. (2024) and Krumpl et al. (2024), with Esaki et al. (2024) also showing that their method outperforms temperature scaling in terms of calibration and OOD detection performance. We note that the method of Esaki et al. (2024) is somewhat more complex than ours, requiring generating synthetic training samples using their Multi-Mixup method. Both methods also require access to some intermediate values of the base neural network model for recalibration, while our method only uses the last-layer logits. Most significantly, these approaches do not explore the impact that homogenization has on OOD detection performance.

Recalibrating Bayesian Models. Most current work in the recalibration of probabilistic models for classification focuses on the standard *point prediction* setting, where probabilities are obtained from a single logit output. This raises the question of how to apply recalibration to *Bayesian* models, which involve the aggregation of multiple intermediate predictions to obtain the probability output. One common example is in dropout neural networks, where a distribution of predictions is produced by repeatedly applying dropout to approximate a Bayesian neural network (Gal and Ghahramani, 2016). In this setting, we cannot directly apply standard recalibration methods that act on vector-valued predictions. Recalibration has not been extensively explored in the Bayesian setting, although Laves et al. (2020b,a) have proposed an approach to extend temperature scaling to dropout predictions.

We show how to extend our method to the Bayesian setting and conduct additional experiments showing that its advantages from the point prediction setting carry over. We introduce three approaches to perform recalibration in this setting. In our first two ap-

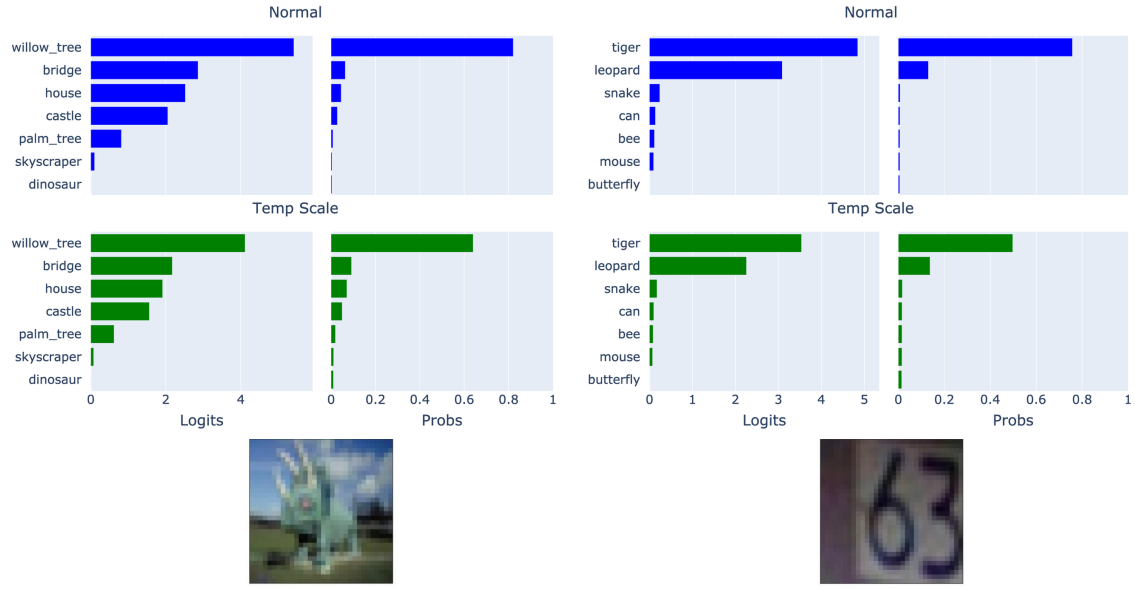

Figure 1: Left: Overconfidence on an in-distribution observation. Right: Overconfidence on an out-of-distribution observation. Temperature scaling reduces overconfidence in both cases.

proaches, we consider two ways to use a point prediction recalibration mapping to transform a distribution of predictions, while in the third approach, we consider converting the Bayesian classifier into a point prediction classifier. We experimentally demonstrate that regardless of the approach, enforcing homogenization allows us to improve the calibration performance of temperature scaling, while retaining the OOD detection performance.

2. Background and Definitions

We consider the multiclass classification setting with C classes. We denote our input space by $\mathcal{X}$ and our output space by $\mathcal{Y} = \{1, \ldots, C\}$. We will specifically consider *probabilistic classifiers*, which return a probability distribution over the classes: $\boldsymbol{q} : \mathcal{X} \to \Delta_C$, where Δ_C is the C-dimensional unit simplex:

$$\Delta_C = \left\{ \boldsymbol{p} \in \mathbb{R}^C : p_1 + \cdots + p_C = 1, p_c \geqslant 0 \text{ for } c \in \{1, \ldots, C\} \right\}.$$

That is, we assume that our classifier returns a probability prediction over the classes, rather than just a single class prediction. In this section, we discuss two ways of evaluating the quality of the probabilistic predictions, as well as the task of improving predictions through recalibration.

2.1 Evaluating Probabilistic Classifiers

How do we evaluate whether or not the predicted probabilities are any good? Modern neural networks are known to suffer from *overconfidence*, predicting high probabilities for misclas-

sified observations (Guo et al., 2017; Lakshminarayanan et al., 2017; Minderer et al., 2021). We can distinguish between overconfidence on *in-distribution* data and overconfidence on *out-of-distribution* data. For example, consider a neural network trained on `CIFAR-100` (Krizhevsky, 2009). In the left part of Figure 1, we show the prediction on a "dinosaur" image, which is one of the classes from `CIFAR-100`; the model is 80% confident that the image is "willow_tree". In the right part of Figure 1, we show the prediction on a StreetView House Numbers (Netzer et al., 2011) (`SVHN`) image; the model is 80% confident that the image is "tiger".

2.1.1 Calibration

First, we discuss the concept known as *calibration*, which is used to evaluate the in-distribution probability predictions of a model. Formally, a classifier $\boldsymbol{q}$ is perfectly calibrated if

$$\mathbb{P}\left(Y = \arg\max_{c \in 1:C} q_c(\boldsymbol{X}) \mid \max_{c \in 1:C} q_c(\boldsymbol{X}) = p\right) = p$$

for all $p \in (0, 1)$ (Guo et al., 2017). In other words, if the predicted probability of the class prediction of a particular observation is p, then probability that we are correct should also be p. Here, we see that overconfidence means that our the accuracy of our predictions will be lower than our predicted probabilities, which can be a significant problem. Hence, most works focus on improving a classifier's calibration. To measure the level of miscalibration, the *expected calibration error* (Naeini et al., 2015) (ECE) is often used (see Section D.5 for details and discussion of related metrics). Note that a model can also be underconfident in some cases as well, but reducing overconfidence is especially important, particularly in high-risk applications (Gawlikowski et al., 2023): in medical image analysis, low-confidence predictions should be given to human experts for further investigation (Kompa et al., 2021), while in object detection for autonomous driving, low-confidence predictions should alert the user to take caution (Feng et al., 2018). In these settings, the consequences of overconfidence are serious.

2.1.2 Out-of-Distribution Detection Performance

Next, we discuss how to evaluate a model's out-of-distribution probabilistic predictions. For an OOD observation, we want the prediction to be as uniform as possible, because it does not correspond to any of the classes. However, we also want to appropriately penalize a classifier that returns a uniform prediction for *all* observations, including in-distribution observations. Accordingly, a standard approach to measuring a model's OOD performance is to evaluate its performance on a binary classification task of predicting whether an observation is in-distribution or out-of-distribution. For example, if we have a model trained on `CIFAR-100`, an OOD detection task could consist of adding an equal number of `SVHN` observations to the test data set. We follow the approach described in Hendrycks and Gimpel (2017), which is a standard approach in the literature (see, e.g., Ren et al., 2019; Liu et al., 2020; Sun et al., 2022, etc.)

Given a classifier $\boldsymbol{q}$, we can define an OOD score function $o : \Delta_C \to \mathbb{R}$, which should return higher values for OOD observations and lower values for in-distribution observations. Some OOD score functions used in the literature include:

- Entropy: $\mathbb{H}(\boldsymbol{p}) = -\sum_{c=1}^{C} p_c \log p_c$ (Ren et al., 2019)
- Variation Ratio[1]: $\text{vratio}(\boldsymbol{p}) = 1 - \max_{c \in 1:C} p_c$. (Hendrycks and Gimpel, 2017)

Then, given an observation $\boldsymbol{x}$, we can define the OOD detection function $d : \mathcal{X} \to \mathbb{R}$ by $d(\boldsymbol{x}) = (o \circ \boldsymbol{q})(\boldsymbol{x})$, which predicts the extent to which an observation $\boldsymbol{x}$ is OOD. To use $d(\boldsymbol{x})$ to predict in/OOD, we would need to specify a cutoff value; however, we can avoid this by computing the AUC of the receiver-operating-curve of the OOD score and the in/OOD labels. In terms of OOD detection performance, an overconfident classifier would mistakenly assign too much probability to an OOD sample, which would decrease the performance.

2.2 Recalibration

The task of improving the probabilistic predictions of a classifier is known as *recalibration*. Most classifiers can be expressed in a two-stage approach

$$\boldsymbol{q} = \boldsymbol{h} \circ \boldsymbol{f}, \quad \boldsymbol{f} : \mathcal{X} \to \mathcal{Z}, \quad \boldsymbol{h} : \mathcal{Z} \to \Delta_C \tag{1}$$

for some *prediction space* $\mathcal{Z}$. First the input $\boldsymbol{x}$ is mapped to an element $\boldsymbol{z} \in \mathcal{Z}$ with an intermediate prediction function $\boldsymbol{f}$, which is then mapped to a probability vector with $\boldsymbol{h}$. In the standard setting, $\mathcal{Z} = \mathbb{R}^C$, and $\boldsymbol{h}$ is the *softmax* function $\boldsymbol{\eta}$:

$$\boldsymbol{\eta}(\boldsymbol{z}) = \left(\frac{\exp(z_1)}{\sum_{c=1}^{C} \exp(z_c)}, \quad \cdots \quad , \quad \frac{\exp(z_C)}{\sum_{c=1}^{C} \exp(z_c)} \right).$$

For example, $\boldsymbol{f}$ could be represented with a neural network that outputs a logit prediction, which then is transformed into a probability vector using the softmax function. In Section 4, we will consider the case where $\mathcal{Z}$ is the set of *cumulative distribution functions* (CDFs) over $\mathbb{R}^C$.

In the task of recalibration, we use a separate validation data set $\mathcal{D}_{\text{val}} = \{(\boldsymbol{x}_i, y_i)\})_{i=1}^{N}$ to learn a *recalibration mapping* $\boldsymbol{g} : \mathcal{Z} \to \mathcal{Z}$ to apply to our predictions. After applying the recalibration mapping $\boldsymbol{g}$, the recalibrated classifier becomes:

$$\tilde{\boldsymbol{q}} = \boldsymbol{h} \circ \boldsymbol{g} \circ \boldsymbol{f}.$$

In order to learn $\boldsymbol{g}$, we identify a parametric class of calibration functions $\boldsymbol{g}_\theta(\boldsymbol{z})$ for some parameter space Θ, and a loss function ℓ. We compute the logit predictions $\boldsymbol{z}_i = \boldsymbol{f}(\boldsymbol{x}_i)$ on $\mathcal{D}_{\text{val}}$, and we solve the optimization problem

$$\underset{\theta \in \Theta}{\arg\min} \quad \sum_{i=1}^{N} \ell\Big((\boldsymbol{h} \circ \boldsymbol{g}_\theta)(\boldsymbol{z}_i), y_i \Big). \tag{2}$$

Typically, ℓ is set to be the cross-entropy loss, and gradient based methods are used to perform the optimization.

1. The term "variation ratio" is taken from Gal et al. (2017).

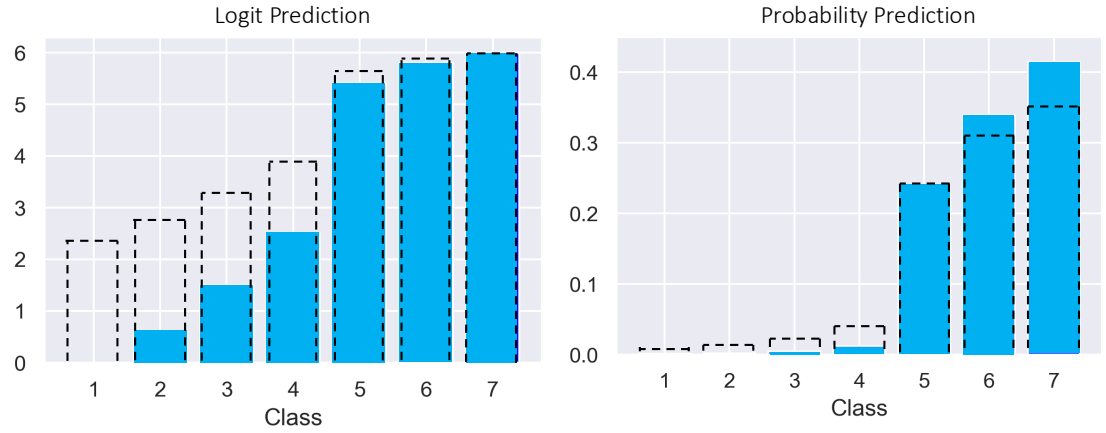

Figure 2: The result of applying temperature scaling on the logit prediction and corresponding effect on the probability prediction. The dotted lines represent the value after temperature scaling. Left: The differences between each class decreases in the logit scale. Right: The classes are divided into two groups; one group that increases in probability and one group that decreases in probability.

2.3 Temperature Scaling

Introduced by Guo et al. (2017), temperature scaling is one of the simplest recalibration methods. We learn a single parameter $T > 0$, which we use to rescale the logit predictions: the function $\boldsymbol{g}$ is of the form

$$\boldsymbol{g}(\boldsymbol{z}) = \begin{bmatrix} T \cdot z_1 & \cdots & T \cdot z_C \end{bmatrix}.$$

When $T < 1$, the logits are scaled closer to 0, which has the effect of making the output probability more uniform. This helps to reduce the degree of overconfidence in the classifier. In the bottom portion of Figure 1, we see that temperature scaling decreases the overconfidence in both the in-distribution and out-of-distribution input image. The main drawback to temperature scaling is its lack of flexibility, since it only learns one parameter to linearly scale the logits. Additional flexibility can be added in several ways, such as by learning a more complex function as in Rahimi et al. (2020) or learning an input-dependent temperature as in Joy et al. (2023), and can result in substantial improvement to calibration.

Despite its simplicity, temperature scaling has some advantages over competing calibration methods. There have been many more flexible methods proposed, many of which learn entire neural network functions. These methods are more time-consuming and difficult to optimize, yet it remains difficult to consistently outperform temperature scaling (see, for example, the results in Rahimi et al., 2020, which shows that temperature scaling performs quite well on some data sets and architectures, even compared to much more sophisticated methods). We can hypothesize that one reason for this is that due to the simplicity of temperature scaling, it very rarely overfits the data. Hence, in our approach, we aim to retain much of the simplicity of temperature scaling.

3. Recalibration by Enforcing Homogenization

In this section, we introduce our approach to recalibration based on constraining the slope of a more flexible class of functions than the linear functions used in temperature scaling. We introduce a process that we term *homogenization* which describes how a probability vector becomes more uniform after applying temperature scaling, and we show that enforcing homogenization implies that the entropy and variation ratio also increases. We then introduce a slope-constrained family of softplus functions for recalibration which enforce homogenization in the recalibrated probability.

To begin, we visually inspect the effect that temperature scaling has on the probability prediction in Figure 2 when the temperature is less than 1. The left plot shows the logit prediction, the right plot shows the probability prediction, and the dotted lines show the temperature scaled prediction. Given a logit vector $\boldsymbol{z}$ with corresponding probability $\boldsymbol{p} = \boldsymbol{\eta}(\boldsymbol{z})$, let us denote the recalibrated logit by $\tilde{\boldsymbol{z}} = T\boldsymbol{z}$ and the recalibrated probability by $\tilde{\boldsymbol{p}} = \boldsymbol{\eta}(\tilde{\boldsymbol{z}})$. In what sense is $\tilde{\boldsymbol{p}}$ more uniform than $\boldsymbol{p}$? In Figure 2, we can see that temperature scaling contracts the input logit vector in the following way:

$$|\tilde{z}_i - \tilde{z}_j| \leqslant |z_i - z_j|, \quad \text{for all } i, j \in \{1, \dots, C\}\,. \tag{3}$$

That is, all of the differences between each component of the logit vector decrease. However, the same cannot be said for the resulting probability vector:

$$|\tilde{p}_i - \tilde{p_j}| \not\leqslant |p_i - p_i|, \quad \text{for all } i, j \in \{1, \dots, C\}\,.$$

For example, we see that the difference in probability between class 4 and class 3 actually increases after temperature scaling. Accordingly, we introduce a weaker notion comparing the uncertainty between two probability vectors.

3.1 The Homogenization Property

Given a probability vector $\boldsymbol{p}$, one basic way of making it more "uniform" is to take some of the probability mass from a class with a lot of probability, and add it to a class with a little probability. A slightly more general procedure is to divide the classes into two groups: one with high probability mass, and one with low probability mass, and to take probability mass away from the first group, and transfer it to the second group. We will refer to this process as *homogenization*. In the following definition, we define a probability vector to be *more homogenous* than another if we can arrive at the probability vector from this process.

Definition 1 *Given $\tilde{\boldsymbol{p}}, \boldsymbol{p} \in \Delta_C$, we say that $\tilde{\boldsymbol{p}}$ is **more homogenous** than $\boldsymbol{p}$ if there exists $k \in \{1, \dots, C\}$ such that for the sets S_1, S_2 defined by*

$$S_1 = \big\{c \in \{1, \dots, C\} \mid p_c \leqslant p_k\big\}, \quad S_2 = \big\{c \in \{1, \dots, C\} \mid p_c > p_k\big\}$$

the following conditions hold:

1. *For all $c \in S_1$, we have $\tilde{p}_c \geqslant p_c$ and for all $c \in S_2$ we have that $\tilde{p}_c < p_c$.*
2. *For all $c \in S_1$ and $c' \in S_2$, we have that $\tilde{p}_c < \tilde{p}_{c'}$.*

That is, there is some cutoff p_k that divides the probability components into two groups. In the group less than or equal to the cutoff, the recalibrated probability increases; in the group greater than the cutoff, the recalibrated probability decreases. The second condition states that after the probability transfer, the larger probability group still has larger probability than the smaller probability group.

In Figure 2, the probability vector after applying temperature scaling is more homogenous than the original probability vector: we took probability mass from classes 6 and 7 and distributed it among classes 1 through 5. This property turns out to be a somewhat strong property in comparing the uniformity of two probability vectors. Specifically, we can show that it is also related to the change in entropy and variation ratios:

Proposition 2 *If $\tilde{\boldsymbol{p}} \in \Delta_C$ is more homogenous than $\boldsymbol{p}$, then both $\mathbb{H}(\tilde{\boldsymbol{p}}) \geqslant \mathbb{H}(\boldsymbol{p})$ and* $\text{vratio}(\tilde{\boldsymbol{p}}) \geqslant \text{vratio}(\boldsymbol{p})$.

The proof is given in Section A.2 of the Appendix. The general idea behind the proof of how the entropy increases is as follows: we know that by transferring a small amount of probability mass from a higher probability class to a lower probability class will increase the entropy. If $\tilde{\boldsymbol{p}}$ is more homogenous than $\boldsymbol{p}$, we can decompose the difference into a sequence of individual probability transfers, each of which increases the entropy.

Note also, that the reverse statements do not necessarily hold; it is possible for the entropy or variation ratio to increase when $\tilde{\boldsymbol{p}}$ is not more homogenous than $\boldsymbol{p}$; we list some examples in Section B.1 of the Appendix. The fact that the entropy and variation ratio is guaranteed to increase plays a significant role in OOD detection using either metric, since we want these metrics to increase on OOD data.

3.2 Enforcing Homogenization in Recalibration

In the previous section, we introduced a formal definition to describe in what sense temperature scaling increases the uniformity of the original probability output. In this section, we will describe how this property can be enforced in recalibration mappings. This will enable us to propose a recalibration method that always increases the uniformity of the probability predictions in the same sense as temperature scaling. First, we introduce the *order-preserving* property (Rahimi et al., 2020): A mapping $\boldsymbol{g} : \mathbb{R}^C \to \mathbb{R}^C$ is **order-preserving** if given any input logit vector $\boldsymbol{z} \in \mathbb{R}^C$ and indices $i, j \in \{1, \ldots, C\}$, we have

$$g_i(\boldsymbol{z}) \leqslant g_j(\boldsymbol{z}) \iff z_i \leqslant z_j.$$

This is an important property for a recalibration mapping to hold because it ensures that the predictions do not change. In Definition 1, this property ensures that the second condition is satisfied. Next, based on (3), we first introduce a definition to compare the extent that two recalibration mappings increase the uniformity of the input prediction.

Definition 3 *Given two recalibration mappings $\boldsymbol{g}^{(1)}, \boldsymbol{g}^{(2)} : \mathbb{R}^C \to \mathbb{R}^C$, we say that $\boldsymbol{g}^{(1)}$ is **more homogenizing** than $\boldsymbol{g}^{(2)}$ if, given any $\boldsymbol{z} \in \mathbb{R}^C$ and pair of indices $i, j \in \{1, \ldots, C\}$, we have*

$$|g_i^{(1)}(\boldsymbol{z}) - g_j^{(1)}(\boldsymbol{z})| \leqslant |g_i^{(2)}(\boldsymbol{z}) - g_j^{(2)}(\boldsymbol{z})|.$$

As a special case of this definition, if $\boldsymbol{g}$ is more homogenizing than the identity function, then we just say that $\boldsymbol{g}$ is *homogenizing*.

Definition 4 *A recalibration mapping $\boldsymbol{g} : \mathbb{R}^C \to \mathbb{R}^C$ is **homogenizing** if it is more homogenizing than the identity function,* $\mathrm{id} : \mathbb{R}^C \to \mathbb{R}^C$ *given by* $\mathrm{id}(\boldsymbol{z}) = \boldsymbol{z}$.

We note that the degree to which a mapping homogenizes the logit vectors may be more relevant than just the fact that homogenization is satisfied. Since our goal is to improve upon temperature scaling, we can consider two approaches: learning a recalibration mapping that is homogenizing, and learning a recalibration mapping that is more homogenizing than temperature scaling. In our experiments in Section 3.4, we show that both have important consequences on the OOD detection performance.

Now, if $\boldsymbol{g}^{(1)}$ is more homogenizing than $\boldsymbol{g}^{(2)}$, we can show that given any input logit vector $\boldsymbol{z}$, the softmax probability returned by $\boldsymbol{g}^{(1)}$ is more homogenous than the softmax probability return by $\boldsymbol{g}^{(2)}$.

Proposition 5 *Given order-preserving functions $\boldsymbol{g}^{(1)}, \boldsymbol{g}^{(2)} : \mathbb{R}^C \to \mathbb{R}^C$, suppose $\boldsymbol{g}^{(1)}$ is more homogenizing than $\boldsymbol{g}^{(2)}$. Then for all $\boldsymbol{z} \in \mathbb{R}^C$, $\tilde{\boldsymbol{p}}^{(1)} = (\boldsymbol{\eta} \circ \boldsymbol{g}^{(1)})(\boldsymbol{z})$ is more homogenous than $\tilde{\boldsymbol{p}}^{(2)} = (\boldsymbol{\eta} \circ \boldsymbol{g}^{(2)})(\boldsymbol{z})$.*

The proof is given in Section A.3 of the Appendix. By setting $\boldsymbol{g}^{(2)}$ to be the identity function, this result states that a homogenizing recalibration mapping always increases the uncertainty in the probability output in terms of homogenization. Now, for recalibration mappings that are not homogenizing, this means that there exists some input $\boldsymbol{z}$ such that the recalibrated probability is not more homogenous than the original probability. It is still possible that for all test observations, we never see this occur, although we find in our experiments that there usually are some observations for which this occurs. Likewise, there are some observations for which the entropy decreases or the variation ratio decreases.

3.3 Enforcing Homogenization by Constraining the Slope

In this section, we introduce our approach to recalibration based on using a family of softplus functions in the optimization (2). To begin, we describe our approach to learning homogenizing recalibration mappings. Then, we describe our approach to learning mappings that are more homogenizing than temperature scaling. In the previous section we showed that this guarantees that the probability predictions become more homogenous in the sense of Definition 1, which intuitively should be desirable in OOD detection.

3.3.1 Homogenization in Diagonal Functions

First, let us identify one more property that we want our recalibration mappings to satisfy, which is known as the diagonal property (Rahimi et al., 2020): A function $\boldsymbol{g} : \mathbb{R}^C \to \mathbb{R}^C$ is **diagonal** if it has the form

$$\boldsymbol{g}(\boldsymbol{z}) = \begin{bmatrix} g_1(\boldsymbol{z}) & \cdots & g_C(\boldsymbol{z}) \end{bmatrix} = \begin{bmatrix} \overline{g}(z_1) & \cdots & \overline{g}(z_C) \end{bmatrix}$$

for some univariate function $\overline{g} : \mathbb{R} \to \mathbb{R}$. This property serves as *regularization*, limiting the class of functions that we can select, preventing overfitting.

In order for a diagonal function to be order-preserving, we need $\overline{g}$ to be an increasing function. In order to additionally be homogenizing, we first note the following equivalence if $\overline{g}$ is increasing: given $\boldsymbol{z} \in \mathbb{R}^C$,

$$\begin{aligned} & |\overline{g}(z_i) - \overline{g}(z_j)| \leqslant |z_i - z_j| && \text{for all } i, j \in \{1, \dots, C\} \\ \iff & \overline{g}(z_i) - \overline{g}(z_j) \leqslant z_i - z_j && \text{for all } i, j \in \{1, \dots, C\} \text{ such that } z_i > z_j \\ \iff & \overline{g}(z_i) - z_i \leqslant \overline{g}(z_j) - z_j && \text{for all } i, j \in \{1, \dots, C\} \text{ such that } z_i > z_j. \end{aligned} \tag{4}$$

Thus, we observe that $\overline{g}$ will be homogenizing if the function $z \mapsto \overline{g}(z) - z$ is decreasing. For differentiable $\overline{g}$, this implies that $g' \leqslant 1$. Since $\overline{g}$ must also be increasing, we have that $0 < g' \leqslant 1$.

3.3.2 Homogenizing Softplus Recalibration

Now, these constraints naturally suggest the application of the *sigmoid* function, in order to squash the output of the slope to be between 0 and 1. That is, we model the slope as a function of the form

$$\overline{g}'(z) = \sigma(l(z)), \quad \sigma(z) = \frac{1}{1 + \exp(-z)}.$$

Here, the function l can be specified to be as complex as desired. For example, l can even be modeled with a neural network, and we could learn the parameters using the approach in Wehenkel and Louppe (2019), as described by Rahimi et al. (2020). However, for our purposes, we elect to use the simplest choice for l: a linear function. We define $l(z; a, b) = -a \cdot (z - b)$, where a determines how quickly the function changes from 0 to 1, and b determines where the change occurs (the negative sign means that for $a > 0$, the sigmoid function starts at 1 and lowers to 0). This choice for the slope results in a simple form for the actual function:

$$\overline{g}(z; a, b) = -(1/a) \log(1 + \exp(-a(z - b))).$$

We can recognize this as a variant of the *softplus* activation function commonly used in neural network architectures. It also has been used in recalibration methods as well, such as in Balanya et al. (2024), although not directly applied to the logits as in our case.

Given logit predictions $\boldsymbol{z}_1, \dots, \boldsymbol{z}_N$ on the validation data set $\mathcal{D}_{\text{val}}$, using the class of softplus functions for recalibration in the optimization problem (2) corresponds to the following homogenizing recalibration mapping:

$$a^*, b^* = \underset{a>0, b \in \mathbb{R}}{\arg\min} \sum_{i=1}^{N} \ell(\boldsymbol{\eta}(\boldsymbol{g}(\boldsymbol{z}_i; a, b)), y_i), \quad \overline{g}^*(z) = \overline{g}(z | a^*, b^*).$$

We call this the *basic softplus* method (SP-1).

3.3.3 Scaling the Sigmoid Function

In order to assess the impact of the homogenization property, we will consider a variation of the approach described above where we apply a *scaled* sigmoid function to the output:

$$\overline{g}'(z; a, b, c) = c \cdot \sigma(l(z; a, b)), \quad c > 0 \tag{5}$$

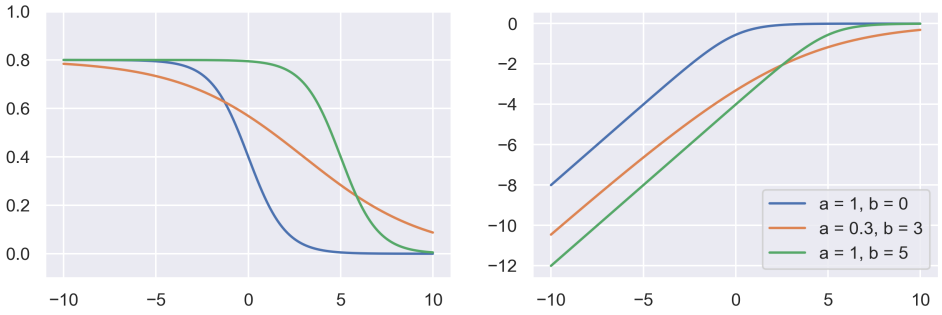

Figure 3: Three different sigmoid functions to model the slope and the corresponding softplus function.

which corresponds to the scaled version of the softplus function:

$$\overline{g}(z, a, b, c) = -(c/a)\log(1 + \exp(-a(z - b))).$$

This adds a new learnable scale parameter c. We call recalibration with this class of functions the *unconstrained softplus* (SP-U) method, and corresponds to the following learned recalibration mapping:[2]

$$a^*, b^*, c^* = \underset{a>0, b\in\mathbb{R}, c>0}{\arg\min} \sum_{i=1}^{N} \ell(\boldsymbol{\eta}(\boldsymbol{g}(\boldsymbol{z}_i; a, b, c)), y_i), \quad \overline{g}^*(z) = \overline{g}(z|a^*, b^*, c^*).$$

We plot a few examples of different softplus functions in Figure 3. The general shape bears a strong resemblance to the increasing function learned in Rahimi et al. (2020), with a fraction of the complexity. Observe that modeling the slope with a scaled sigmoid function constitutes an extension to temperature scaling. Given a temperature T, we can approximate the linear temperature scaling function by setting $c = T$ and setting b to be arbitrarily large. In Figure 4, we show visually how increasing the value of b causes the function to become closer to a linear function.

If c is learned to be greater than 1, this results in a recalibration mapping that is not homogenizing. In Section 3.4, we explore the impact that adding this parameter has on the calibration metrics and the OOD detection metrics. Intuitively, we might expect that the increased flexibility improves the calibration metrics, but the lack of homogenization worsens the OOD detection metrics.

3.3.4 Using Temperature Scaling to Constrain the Slope

We motivated modeling the slope with a sigmoid function with the idea of enforcing the homogenization property. However, it's possible to enforce this property in a *minimal* way. For example, it is possible that our learned mapping always increases the homogenization,

2. Note that we are overloading the definition of $\overline{g}$; to make this consistent with the definition of $\overline{g}$ from the previous section, we adopt the convention that $c = 1$ by default if it is omitted as an argument.

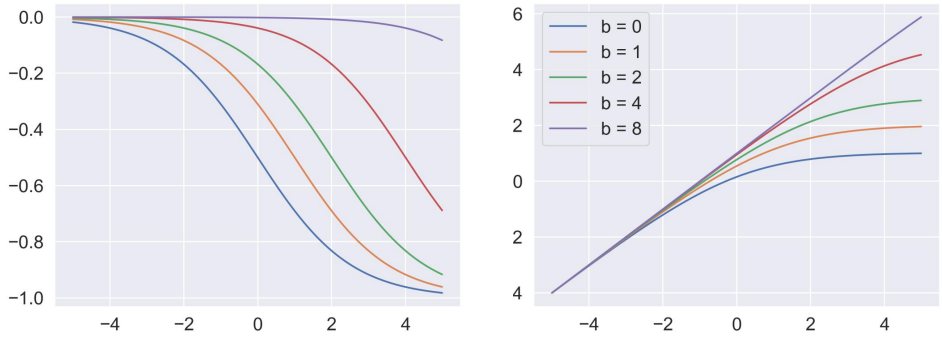

Figure 4: Plots of sigmoid and corresponding softplus functions with $a = 1, c = 0.8$, and different values of b. Note that we add a constant to each function so that the leftmost values are equal for each (adding a constant does not change the recalibrated probabilities since the softmax function is applied.)

but only by a negligible amount. In that case, this mapping would not be particularly useful in reducing our base model's overconfidence. In this section, we consider learning a mapping that increases the homogenization *more* than temperature scaling (Definition 3). Given a temperature scaling map with temperature T, using the same logic as in the discussion of (4), a diagonal and order-preserving function will be more homogenizing than the temperature scaling map if and only if the function $z \mapsto \overline{g}(z) - Tz$ is decreasing; for differentiable g, this means $0 < \overline{g}'(z) \leqslant T$. Accordingly, our recalibration method is as follows: we first find the optimal temperature using temperature scaling:

$$T^* = \underset{T>0}{\arg\min} \sum_{i=1}^{N} \ell(\boldsymbol{\eta}(T\boldsymbol{z}_i), y_i) \tag{6}$$

and we fix $c = T^*$ in (5). Then, we optimize over a and b. We call this recalibration method *temperature scaled softplus* (SP-TS) and it corresponds to the following learned mapping:

$$a^*, b^* = \underset{a>0, b\in\mathbb{R}}{\arg\min} \sum_{i=1}^{N} \ell(\boldsymbol{\eta}(\boldsymbol{g}(\boldsymbol{z}_i; a, b, T^*)), y_i), \quad \overline{g}^*(z) = \overline{g}(z|a^*, b^*, T^*).$$

Since $\boldsymbol{g}$ is more homogenizing than temperature scaling, from Proposition 5, this implies that in the probability output, the entropy and variation ratio of the probability prediction is also greater than temperature scaling. Additionally, as b increases, the solution converges to the temperature scaling solution, as illustrated in Figure 4. In Section 3.4, we compare this approach with temperature scaling, and find that it results in improved calibration with comparable OOD detection performance.

Note that in the optimization (6), we do not constrain $T \leqslant 1$, which means we could have $T^* > 1$, which corresponds to a temperature scaling mapping that is not homogenizing. In this case, the SP-TS recalibration mapping would still be more homogenizing than TS, but not more so than the identity mapping (Definition 4). However, experimentally, we found that this never occurred; due to the typical overconfidence of the trained models, T^* was always less than 1.

3.4 Evaluation on CIFAR-100

In this section, we will evaluate our proposed softplus recalibration methods, as well as temperature scaling, on various deep learning models trained on the CIFAR-100 data set. Our goal is to demonstrate the effectiveness of our methods by showing that using the more flexible softplus functions enables us to improve the calibration of the resulting classifier to a greater extent than temperature scaling. Additionally, we want to show that the homogenization property of temperature scaling is important for OOD detection performance, and placing suitable constraints on the slope of the softplus functions is beneficial for the OOD detection performance. To do so, we also compare our methods to the *unconstrained* SP-U recalibration.

We use the standard training split of CIFAR-100 to train five models[3]: ResNet (He et al., 2016), SE-Net (Hu et al., 2018), DenseNet (Huang et al., 2017), Xception (Chollet, 2017), and Inception (Szegedy et al., 2017). The standard test split of CIFAR-100 consists of 10000 observations. We randomly sample 8000 observations to create the validation data set $\mathcal{D}$, which we will use to learn the recalibration mappings for each method. We perform the optimization using gradient descent, which is automatically implemented using PyTorch. For more implementation details, see Section C.3 of the Appendix.

We use the remaining 2000 observations for testing. Next, we introduce the metrics that we use to evaluate the in-distribution performance of the recalibrated classifiers. We compute the ECE with 10 bins to measure calibration, and we also report the OOD detection performance by measuring the AUC when using each of the classifiers for OOD detection. Specifically, we consider two experiments: in the first, we add 2000 observations from the SVHN data set to the test data set, and for the second, we add 2000 observations from the classroom split of the LSUN (Yu et al., 2015) data set to the test data set. For the OOD detection function, we will use the entropy, as we find that it produces the best results in general; in Section D.4 of the Appendix, we also show the results when using the variation ratio.

3.4.1 Results

We display the results in Figure 5. First, we observe in the top portion that each base model has extremely poor calibration, with ResNet performing the worst. Using any recalibration method results in dramatic improvement. In terms of OOD detection, we find that base models also perform poorly, although the differences in performance with the recalibrated models is not as large in magnitude. In ResNet, SE-Net, and DenseNet, all recalibration methods improve the AUC. However, with Inception, SP-U actually results in a slightly lower AUC than the base method. The fact that SP-U, the only method that is not constrained to be homogenizing, is also the only method that sometimes decreases the OOD detection performance supports the principle that homogenization is beneficial for OOD detection.

Next, we turn to the bottom portion, which provides a closer look at the performance of each of the recalibration methods. We observe that temperature scaling universally has the highest ECE on each of the models. This suggests that the increased flexibility of the

3. The code for the implementation and training of the models is taken from https://github.com/weiaicunzai/pytorch-cifar100.

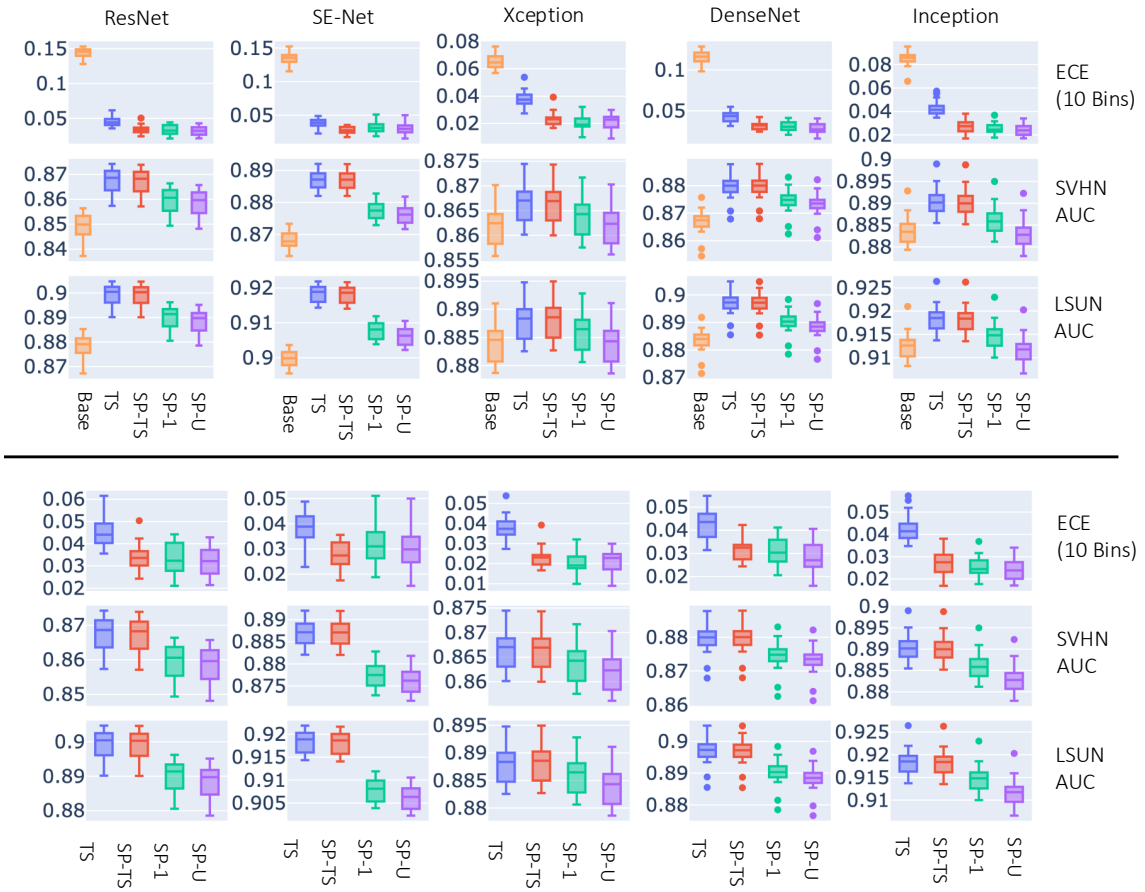

Figure 5: Evaluation metrics for each method and neural network model on `CIFAR-100`. In the bottom set of plots, we omit the results from the base model to more easily see the differences between each method.

	Avg. Entropy Difference					Avg. % Entropy Decrease				
Method	Res	SE	Xcept	Dense	Incept	Res	SE	Xcept	Dense	Incept
TS	1.102	0.958	0.416	0.835	0.478	0.0	0.0	0.0	0.0	0.0
SP-TS	1.122	0.977	0.444	0.857	0.496	0.0	0.0	0.0	0.0	0.0
SP-1	0.682	0.542	0.201	0.47	0.224	0.0	0.0	0.0	0.0	0.0
SP-U	0.63	0.493	0.009	0.386	0.037	0.0	0.0	0.597	0.023	0.575

Table 1: In the Avg. % Entropy Decrease column, we show the proportion of observations on `SVHN` that see a decrease in entropy after recalibration. In the Avg. Entropy Difference column, we show the average difference in entropy of the predictions after applying recalibration (a positive number means the average entropy increases).

softplus methods allows for improved calibration. We find that each of the softplus methods results in similar improvement in ECE. Interestingly, although SP-U has the advantage of an additional free parameter, this does not appear to provide any benefit in terms of ECE; simply using the softplus function alone is sufficient. In terms of OOD detection performance, we observe that TS and SP-TS have the highest AUC, with nearly equal performance. Meanwhile, on both data sets, we see that SP-U and SP-1 both have significantly lower AUCs than SP-TS. This suggests that the degree of homogenization that SP-TS provides is more important than just enforcing homogenization in the SP-1 method. Nevertheless, SP-1 always has a small advantage over SP-U in terms of AUC, with the largest difference on Inception. These results support the conclusion that the homogenization property in general is important in OOD detection performance. We find that SP-TS strikes the best balance between the goals of calibration and OOD detection performance in terms of ECE and AUC, respectively.

Impact of Homogenization. Next, let us analyze the impact of the homogenization property in more detail by examining how each method changes the entropy of the `SVHN` predictions in Table 1 (see Section D.1 for the corresponding table for `LSUN`). First, we note that of all the methods, SP-U was the only one that can result in a decrease in entropy. This is reflected in the "Avg. % Entropy Decrease" column of Table 1. On Xception and Inception, this proportion is quite large, while it's small on DenseNet, and 0 on ResNet and SE-Net. In the "Avg. Entropy Difference" column, we show the difference in average entropy after recalibration. In most cases, recalibration will increase the average entropy, but we can see that SP-U results in the smallest change in entropy. SP-1 has a larger change, followed by TS, and then SP-TS. This supports the conclusion that the degree of homogenization plays a large role in OOD detection performance, which explains why SP-TS performs much better than SP-1.

However, we also note that SP-TS, being more homogenizing than TS, also increases the entropy of the OOD observations more than TS, but we do not see an improvement in AUC. This may be because the average entropy difference metric does not perfectly correlate to improved AUC. In this case, it appears that the additional entropy increase produced by SP-TS does not actually help separate additional OOD observations from in-distribution observations. Future work should investigate approaches to make the additional entropy increase "sharper" to help separate the OOD observations.

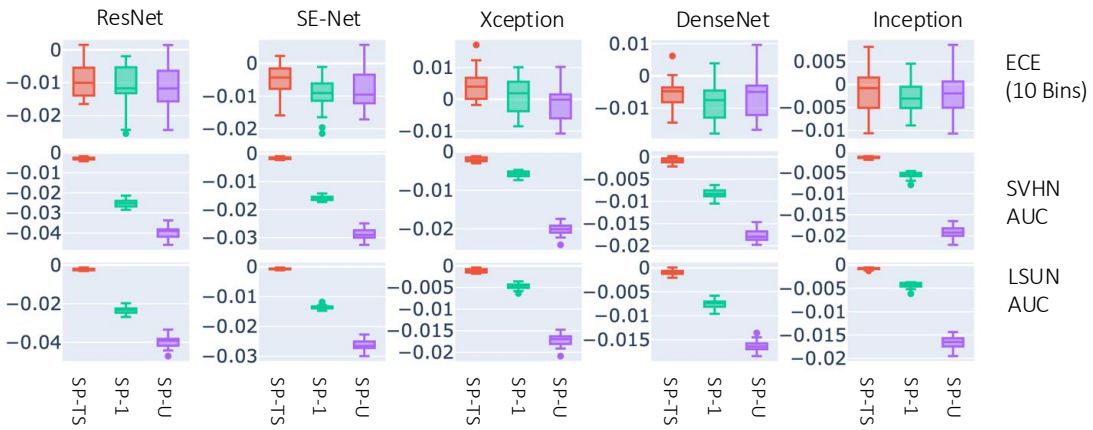

Figure 6: Difference between the evaluation metrics when learning shift parameter and the original evaluation metrics for each method.

Impact on Underconfidence. In this work, we have mainly focused on addressing overconfidence. However, this raises the question of the risk of introducing excessive underconfidence in the predictions, which also may be undesirable. First, in terms of OOD detection performance, underconfidence would manifest itself in a worse AUC due to increasing the uncertainty of in-distribution observations. The results in Figure 5 indicate that this is not an issue for SP-TS. Despite increasing the uncertainty of the predictions more than TS, the AUC of both methods are nearly the same on both OOD data sets. In addition, compared to the base model, all recalibration methods have improved OOD detection performance, except for SP-U with Inception. In terms of calibration, underconfidence refers to the predicted probabilities being lower than the true accuracy. We show additional evaluation metrics to assess the degree of underconfidence introduced in Section D.5 of the Appendix. Our results show that SP-TS does result in increased underconfidence compared to temperature scaling, albeit to a small extent. This should be taken into consideration if underconfidence is highly discouraged. In this case, SP-1 offers a balance between the two, with lower overconfidence than TS and lower underconfidence than SP-TS.

3.4.2 Performance after Adding Complexity

The homogenizing property generally helps to improve the OOD detection ability of the model, although ensuring greater homogenization than temperature scaling is more beneficial than just ensuring homogenization. We note that homogenization is a general property that can be satisfied by any recalibration mapping. In this section, we investigate the effect of homogenization on a slightly more complex suite of recalibration mappings, in which we find that it can protect significantly against degradation of OOD detection performance.

Unlike simple temperature scaling, our softplus recalibration mappings are not invariant to location shifts in the input logit vector. That is, given a logit vector $\boldsymbol{z} \in \mathbb{R}^C$ and a scalar $a \in \mathbb{R}$, we have that in general, $\boldsymbol{\eta}(\boldsymbol{g}(\boldsymbol{z} + a)) \neq \boldsymbol{\eta}(\boldsymbol{g}(\boldsymbol{z}))$. One simple way to add flexibility

	Avg. Entropy Difference					Avg. % Entropy Decrease				
Method	Res	SE	Xcept	Dense	Incept	Res	SE	Xcept	Dense	Incept
SP-TS	1.125	0.974	0.443	0.863	0.496	0.0	0.0	0.0	0.0	0.0
SP-1	0.604	0.499	0.185	0.464	0.203	0.0	0.0	0.0	0.0	0.0
SP-U	0.485	0.393	-0.069	0.328	-0.055	0.106	0.127	0.651	0.219	0.638

Table 2: Avg. % Entropy Decrease and Entropy Difference when learning additional shift.

to the base method is to learn a value $a(\boldsymbol{z})$ to add to the input logit before applying the recalibration mapping. Specifically, we consider the function $a(\boldsymbol{z}) = \boldsymbol{w}^\top \boldsymbol{z}$ (note that the idea of learning a vector to take an inner product with the input logit is also seen in, for example, Balanya et al., 2024). Given a recalibration mapping $\boldsymbol{g}$, and a vector $\boldsymbol{w} \in \mathbb{R}^C$, we define a new mapping $\tilde{\boldsymbol{g}}$ by

$$\tilde{\boldsymbol{g}}(\boldsymbol{z}) = \boldsymbol{g}\big(z_1 + \boldsymbol{w}^\top \boldsymbol{z},\ \ldots\ , z_C + \boldsymbol{w}^\top \boldsymbol{z}\big).$$

This serves to add more flexibility to the base method by adding an additional learnable parameter $\boldsymbol{w}$. Note that if $\boldsymbol{g}$ is homogenizing, then given $\boldsymbol{z} \in \mathbb{R}^C$ and indices $i, j \in \{1, \ldots, C\}$, we have

$$\begin{aligned} |\tilde{g}_i(\boldsymbol{z}) - \tilde{g}_j(\boldsymbol{z})| &= |g_i(z_i + \boldsymbol{w}^\top \boldsymbol{z}) - g_j(z_j + \boldsymbol{w}^\top \boldsymbol{z})| \\ &\leqslant |z_i + \boldsymbol{w}^\top \boldsymbol{z} - (z_j + \boldsymbol{w}^\top \boldsymbol{z})| \\ &= |z_i - z_j| \end{aligned}$$

so $\tilde{\boldsymbol{g}}$ is also homogenizing. We apply this to our softplus methods to show the effect of enforcing the homogenization property in a method that involves learning more parameters. We apply this to our softplus methods to show the effect of enforcing the homogenization property in a method that involves learning more parameters ($\boldsymbol{w}$ is learned jointly with the original parameters).

We show the results in Figure 6. In this plot, we show the difference between the value of the metric after applying the shift and before applying the shift (TS is not displayed because the shift does not affect the prediction). We can see that adding the shift tends to decrease the ECE of each method, with SP-TS improving the least and SP-U improving the most. However, these differences in improvement are quite small. On the other hand, we see comparatively much larger differences in the degradation of OOD detection performance. On SP-TS, there is almost no change, and the decrease in AUC on SP-U is significantly larger than the decrease in AUC on SP-1. In Table 2, we show the average increase in entropy and the average proportion of observations that decrease in entropy for each method. In comparison to Table 1, we observe that the proportion of observations for which SP-U decrease in entropy increases. Furthermore, the average increase in entropy also decreases. These experiments further demonstrate the importance of the homogenization property on OOD detection performance. Minimally enforcing homogenization helps to limit the decrease in performance, while enforcing the homogenization to be greater than temperature scaling almost reduces it to 0.

4. Extension to the Bayesian Setting

In Section 3, we explored the effects of enforcing homogenization when recalibrating with softplus functions. We found that using the softplus functions improves the ECE over temperature scaling, while enforcing the homogenization property helps to improve the OOD detection performance, being able to match the performance of temperature scaling. In this section, we continue our exploration of the effects of homogenization, now in the Bayesian setting. First, we define the form of a Bayesian classifier using the two stage approach (1). Then, we provide three approaches to extend a point prediction recalibration mapping to Bayesian classifiers. Finally, we repeat our experiments from Section 3 in the Bayesian setting.

4.1 Bayesian Classifiers

In the point prediction setting, the intermediate prediction function $\boldsymbol{f}_{\text{point}}$ takes in an input $\boldsymbol{x} \in \mathcal{X}$ and returns a logit vector $\boldsymbol{z} \in \mathbb{R}^C$, which is transformed into a probability vector using the softmax function. In contrast, in the Bayesian setting, we do not obtain the probability prediction from *one* logit prediction; rather, we obtain it from a *distribution* of logit predictions. Formally, we specify these distributions using their CDF: we have an intermediate prediction function $\boldsymbol{f}_{\text{bayes}} : \mathcal{X} \to \mathcal{Z}$, where $\mathcal{Z}$ is the set of CDFs on $\mathbb{R}^C$. Given an input $\boldsymbol{x}$, we obtain a CDF prediction $F : \mathbb{R}^C \to [0, 1]$ by $F = \boldsymbol{f}_{\text{bayes}}(\boldsymbol{x})$. Equivalently, we can imagine predicting a logit vector $\boldsymbol{z}$, now a *random variable* with distribution F, and to obtain a probability prediction $\boldsymbol{p} \in \Delta_C$, we take the expectation of $\boldsymbol{\eta}(\boldsymbol{z})$:

$$\boldsymbol{p} = \mathbb{E}_{\boldsymbol{z} \sim F}[\boldsymbol{\eta}(\boldsymbol{z})]. \tag{7}$$

In the lens of the two-stage approach (1) from Section 2.2, the probabilistic classifier $\boldsymbol{q} : \mathcal{X} \to \Delta_C$ has the form $\boldsymbol{q} = \boldsymbol{h} \circ \boldsymbol{f}_{\text{bayes}}$, where $\boldsymbol{h} : \mathcal{Z} \to \Delta_C$ is given by $\boldsymbol{h}(F) = \mathbb{E}_{\boldsymbol{z} \sim F}[\boldsymbol{\eta}(\boldsymbol{z})]$.

4.2 Recalibration in the Bayesian Setting

Recalibration in the Bayesian setting can still be formulated using the framework from Section 2.2. We learn a recalibration mapping $\boldsymbol{g}_{\text{bayes}} : \mathcal{Z} \to \mathcal{Z}$, which now is a function that maps a CDF to another CDF, and the recalibrated classifier is given by:

$$\tilde{\boldsymbol{q}} = \boldsymbol{h} \circ \boldsymbol{g}_{\text{bayes}} \circ \boldsymbol{f}_{\text{bayes}}. \tag{8}$$

As a result, the space of potential recalibration mappings is enormous. How can we learn $\boldsymbol{g}_{\text{bayes}}$? In this section, we discuss two approaches to obtain a mapping $\boldsymbol{g}_{\text{bayes}}$, given an ordinary point prediction mapping $\boldsymbol{g}_{\text{point}} : \mathbb{R}^C \to \mathbb{R}^C$ to the Bayesian setting. This will result in a family of Bayesian recalibration mappings that are parameterized in the same way as the point prediction mappings, so we can learn the optimal Bayesian mapping via gradient descent. We also consider a third approach that converts the Bayesian classifier to a point prediction classifier to learn $\boldsymbol{g}_{\text{point}}$ normally.

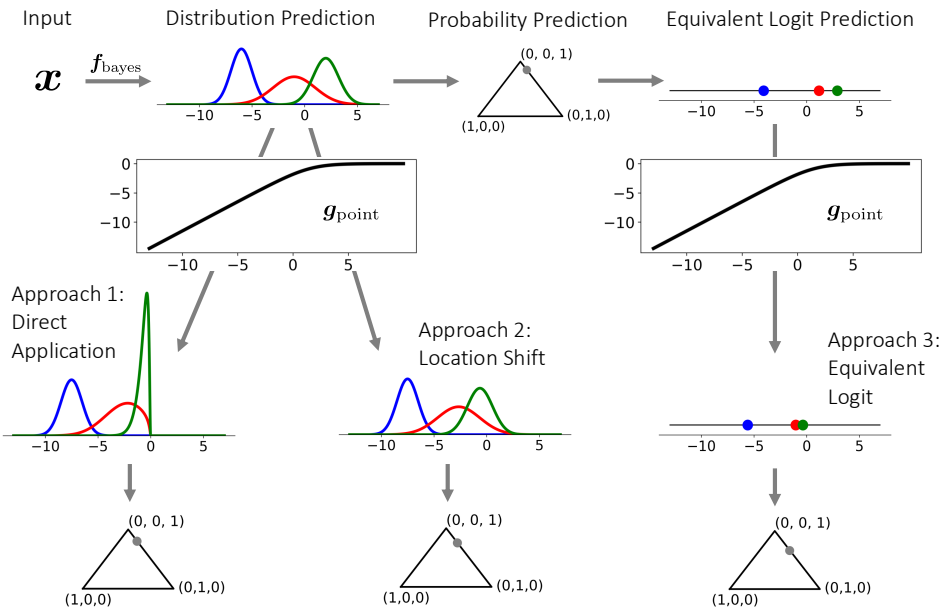

Figure 7: An illustration of each approach to recalibrate a three-component independent Gaussian distribution using a softplus point prediction mapping. We can see that Approach 1 results in a much more complex change in the initial distribution than Approach 2.

4.2.1 Extending Point Prediction Recalibration Mappings

We summarize our three approaches utilizing $\boldsymbol{g}_{\text{point}}$ to obtain the recalibrated probability prediction $\tilde{\boldsymbol{p}}$ as follows:

$$\tilde{\boldsymbol{p}} = \mathbb{E}_{\boldsymbol{z}\sim F}\big[\boldsymbol{\eta}\big(\boldsymbol{g}_{\text{point}}(\boldsymbol{z})\big)\big] \qquad \text{(Approach 1)}$$
$$\tilde{\boldsymbol{p}} = \mathbb{E}_{\boldsymbol{z}\sim F}\big[\boldsymbol{\eta}\big(\boldsymbol{z} - \overline{\boldsymbol{z}} + \boldsymbol{g}_{\text{point}}(\overline{\boldsymbol{z}})\big)\big] \qquad \text{(Approach 2)}$$
$$\tilde{\boldsymbol{p}} = \boldsymbol{\eta}\big(\boldsymbol{g}_{\text{point}}(\overline{\boldsymbol{z}}^{(h)})\big) \qquad \text{(Approach 3)}$$

where $\overline{\boldsymbol{z}} = \mathbb{E}_{\boldsymbol{z}\sim F}[\boldsymbol{z}]$ and $\overline{\boldsymbol{z}}^{(h)} = \boldsymbol{\eta}^{-1}\big(\mathbb{E}_{\boldsymbol{z}\sim F}\big[\boldsymbol{\eta}(\boldsymbol{z})\big]\big)$, with $\boldsymbol{\eta}^{-1}(\boldsymbol{p}) = \log \boldsymbol{p} - \frac{1}{C}\sum_{c=1}^{C} \log p_c$.

In Approach 1, we directly apply $\boldsymbol{g}_{\text{point}}$ to $\boldsymbol{z}$, which is a generalization of the approach from Laves et al. (2020b), originally applied to temperature scaling. In Approach 2, we shift the mean of $\boldsymbol{z}$ from $\overline{\boldsymbol{z}}$ to $\boldsymbol{g}_{\text{point}}(\overline{\boldsymbol{z}})$. From the lens of the two stage framework (8), we can view these two approaches as taking the expectation over $\boldsymbol{z}$ with a transformed CDF $\tilde{F} = \boldsymbol{g}_{\text{bayes}}(F)$: the recalibrated probability prediction is now given by $\tilde{\boldsymbol{p}} = \mathbb{E}_{\boldsymbol{z}\sim\tilde{F}}[\boldsymbol{\eta}(\boldsymbol{z})]$, where we have that

$$\boldsymbol{g}_{\text{bayes}}(F) = F \circ \boldsymbol{g}_{\text{point}}^{-1} \qquad \text{(Approach 1)}$$
$$\boldsymbol{g}_{\text{bayes}}(F) = F\big((\,\cdot\,) + \overline{\boldsymbol{z}} - \boldsymbol{g}_{\text{point}}(\overline{\boldsymbol{z}})\big). \qquad \text{(Approach 2)}$$

Specifically, we have that in Approach 1,

$$\begin{aligned}\tilde{\boldsymbol{p}} &= \mathbb{E}_{\tilde{\boldsymbol{z}}\sim\tilde{F}}[\boldsymbol{\eta}(\tilde{\boldsymbol{z}})] = \mathbb{E}_{\boldsymbol{g}_{\text{point}}(\boldsymbol{z})\sim\tilde{F}}[\boldsymbol{\eta}(\boldsymbol{g}_{\text{point}}(\boldsymbol{z}))] \\ &= \mathbb{E}_{\boldsymbol{g}_{\text{point}}(\boldsymbol{z})\sim F\circ\boldsymbol{g}_{\text{point}}^{-1}}[\boldsymbol{\eta}(\boldsymbol{g}_{\text{point}}(\boldsymbol{z}))] = \mathbb{E}_{z\sim F}[\boldsymbol{\eta}(\boldsymbol{g}_{\text{point}}(\boldsymbol{z}))],\end{aligned}$$

and similarly in Approach 2,

$$\begin{aligned}\tilde{\boldsymbol{p}} &= \mathbb{E}_{\tilde{\boldsymbol{z}}\sim\tilde{F}}[\boldsymbol{\eta}(\tilde{\boldsymbol{z}})] = \mathbb{E}_{\boldsymbol{z}-\overline{\boldsymbol{z}}+\boldsymbol{g}_{\text{point}}(\overline{\boldsymbol{z}})\sim\tilde{F}}[\boldsymbol{\eta}(\boldsymbol{z} - \overline{\boldsymbol{z}} + \boldsymbol{g}_{\text{point}}(\overline{\boldsymbol{z}}))] \\ &= \mathbb{E}_{\boldsymbol{z}-\overline{\boldsymbol{z}}+\boldsymbol{g}_{\text{point}}(\overline{\boldsymbol{z}})\sim F((\cdot)+\overline{\boldsymbol{z}})-\boldsymbol{g}_{\text{point}}(\overline{\boldsymbol{z}})}[\boldsymbol{\eta}(\boldsymbol{z} - \overline{\boldsymbol{z}} + \boldsymbol{g}_{\text{point}}(\overline{\boldsymbol{z}}))] \\ &= \mathbb{E}_{\boldsymbol{z}\sim F}[\boldsymbol{\eta}(\boldsymbol{z} - \overline{\boldsymbol{z}} + \boldsymbol{g}_{\text{point}}(\overline{\boldsymbol{z}}))].\end{aligned}$$

In contrast, in Approach 3, we bypass the task of learning a CDF-valued recalibration mapping altogether by converting the Bayesian classifier into a point prediction classifier. Given a probability prediction $\boldsymbol{p} = \mathbb{E}_{\boldsymbol{z}\sim F}[\boldsymbol{\eta}(\boldsymbol{z})]$, we can find a logit vector $\overline{\boldsymbol{z}}^{(h)} \in \mathbb{R}^C$ that satisfies $\boldsymbol{\eta}(\overline{\boldsymbol{z}}^{(h)}) = \boldsymbol{p}$. In the case of the softmax function, it is invariant to shifting by a constant, so in our selection of the inverse function $\boldsymbol{\eta}^{-1}$, we simply assume that the original logit vector sums to 0 (note that due to the nonlinearity of the softmax function, $\overline{\boldsymbol{z}}^{(h)} \neq \overline{\boldsymbol{z}}$).

Visualization of each Approach. In Figure 7, we visually illustrate each approach in a toy example with $C = 3$ where the base distribution prediction corresponds to an independent normal distribution, which we represent by plotting the three marginal probability density functions. We can see that Approach 1 applies a rather complex transformation to the original distribution of the logits. This is our motivation for introducing Approach 2, which transforms the distribution in a much more interpretable way by keeping the shape the same and only changing the mean. Approach 3 is the simplest because it does not even learn a mapping on the distribution. However, in some applications, the output probability

is not the only quantity of interest; the logit distribution predictions are themselves of some utility, such as in epistemic uncertainty quantification (Hüllermeier and Waegeman, 2021). If Approach 3 is used, there is no corresponding recalibrated logit distribution to obtain these uncertainties. Regardless, with our current set of experiments, this limitation does not manifest itself, since the OOD score function is taken to be the entropy of the final probability vector, so the distribution-level predictions of Approaches 1 and 2 are not needed.

4.2.2 Implementation Using Dropout

One common situation where Bayesian classifiers arise is when we have a neural network trained with dropout, and we use dropout to output multiple predictions for a given test point. We can view the set of predictions as a discrete probability distribution: given a test point $\boldsymbol{x}$, we can use dropout to output L predictions $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_L)$ which corresponds to a discrete distribution with positive probability mass at points $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_L$ and (7) amounts to taking the average of the softmax dropout predictions: $\boldsymbol{p} = \frac{1}{L}\sum_{l=1}^{L} \boldsymbol{\eta}(\boldsymbol{z}_l)$.

To learn the recalibration mappings, we obtain dropout predictions for each observation on the validation set $\mathcal{D}_{\text{val}}$: $\{(\boldsymbol{z}_{i1}, \ldots, \boldsymbol{z}_{iL})\}_{i=1}^{N}$. In the optimization framework (2) of Section 2.2, each approach corresponds to solving the following optimization problems:

$$\underset{\theta\in\Theta}{\arg\min} \quad \sum_{i=1}^{N} \ell\left(\frac{1}{L}\sum_{l=1}^{L}(\boldsymbol{\eta}\circ\boldsymbol{g}_\theta)(\boldsymbol{z}_{il}),\ y_i\right) \tag{Approach 1}$$

$$\underset{\theta\in\Theta}{\arg\min} \quad \sum_{i=1}^{N} \ell\left(\frac{1}{L}\sum_{l=1}^{L}\boldsymbol{\eta}\big(\boldsymbol{z}_{il} - \overline{\boldsymbol{z}}_i + \boldsymbol{g}_\theta(\overline{\boldsymbol{z}}_i)\big),\ y_i\right) \tag{Approach 2}$$

$$\underset{\theta\in\Theta}{\arg\min} \quad \sum_{i=1}^{N} \ell\left((\boldsymbol{\eta}\circ\boldsymbol{g}_\theta)\big(\overline{\boldsymbol{z}}_i^{(h)}\big),\ y_i\right) \tag{Approach 3}$$

where $\overline{\boldsymbol{z}}_i = \frac{1}{L}\sum_{l=1}^{L}\boldsymbol{z}_{il}$ and $\overline{\boldsymbol{z}}_i^{(h)} = \boldsymbol{\eta}^{-1}\left(\frac{1}{L}\sum_{l=1}^{L}\boldsymbol{\eta}(\boldsymbol{z}_{il})\right)$.

Computationally, we can see that Approach 3 has the advantage because we can precompute the equivalent logit vectors beforehand, and the optimization becomes the same as the point prediction setting. In contrast, Approaches 1 and 2 require that we keep all the dropout predictions on hand during the optimization. This can be expensive in terms of memory usage, especially if a large number of dropout samples is used. In this case, it may be necessary to load the dropout samples in batches in each iteration. Between Approaches 1 and 2, Approach 2 has the advantage in computational efficiency because the recalibration mapping only needs to be applied to the mean dropout logit vector in each iteration, while in Approach 1, the recalibration mapping needs to be applied to *all* of the dropout logit vectors. This becomes more significant as the recalibration mapping increases in complexity.

4.3 Evaluation on `CIFAR-100`

We apply the experiment from Section 3.4 to each of the three approaches. Note that in each approach from Section 4.2, $\boldsymbol{g}_{\text{bayes}}$ has the same parameterization as $\boldsymbol{g}_{\text{point}}$, so the optimization process is done in the same way using gradient descent. For each deep learning

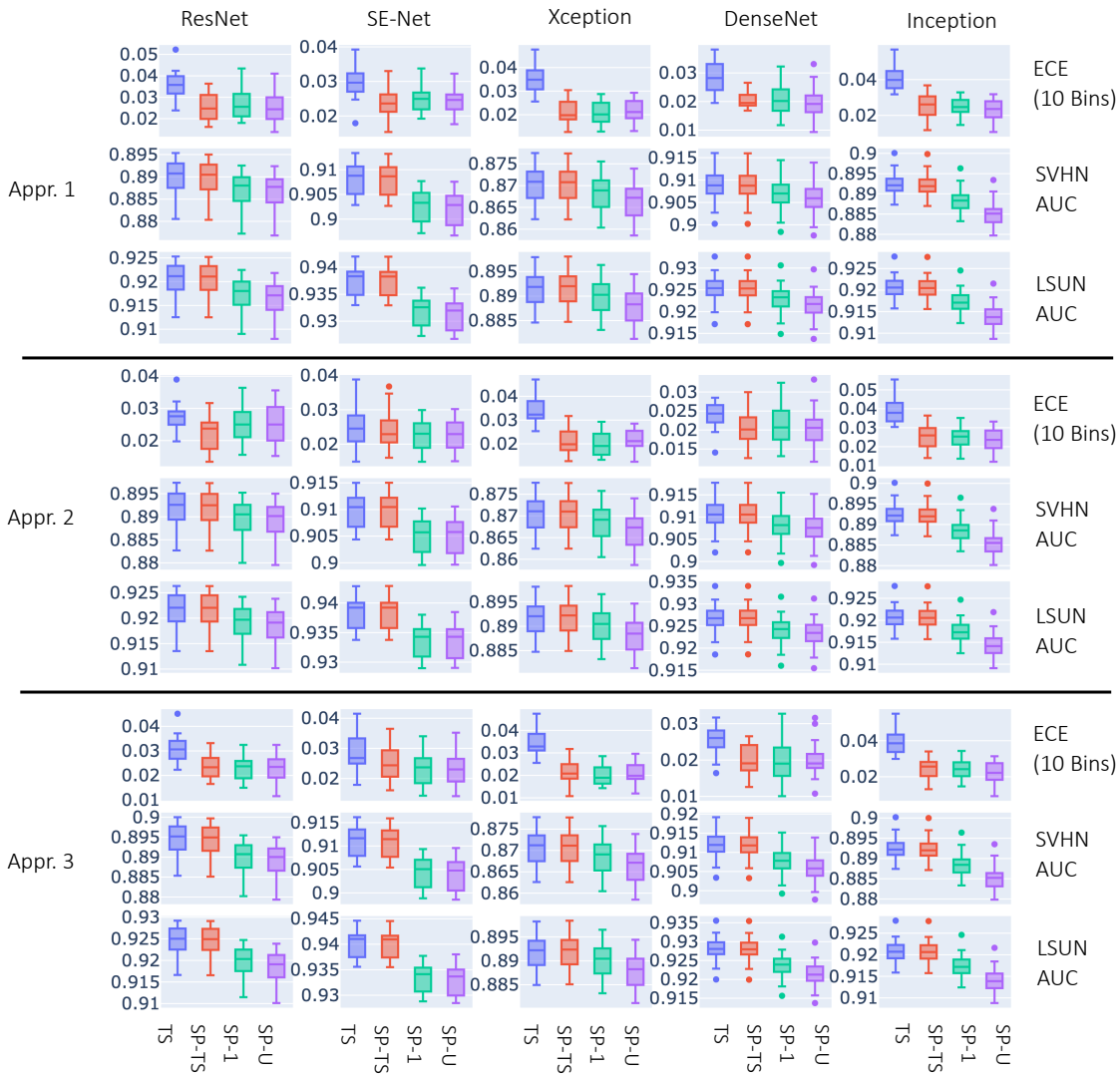

Figure 8: Evaluation metrics for each of the three approaches to recalibration in the Bayesian setting.

		Avg. Entropy Difference					Avg. % Entropy Decrease				
Type	Method	Res	SE	Xcept	Dense	Incept	Res	SE	Xcept	Dense	Incept
Appr. 1	TS	0.659	0.574	0.342	0.356	0.427	0.0	0.0	0.0	0.0	0.0
Appr. 1	SP-TS	0.677	0.59	0.369	0.372	0.444	0.0	0.0	0.0	0.0	0.0
Appr. 1	SP-1	0.357	0.266	0.159	0.15	0.194	0.0	0.0	0.0	0.0	0.0
Appr. 1	SP-U	0.292	0.244	-0.048	0.055	0.0	0.042	0.0	0.685	0.348	0.636
Appr. 2	TS	0.64	0.563	0.346	0.361	0.424	0.0	0.0	0.0	0.0	0.0
Appr. 2	SP-TS	0.647	0.571	0.37	0.367	0.44	0.0	0.0	0.0	0.0	0.0
Appr. 2	SP-1	0.376	0.289	0.162	0.154	0.195	0.0	0.0	0.0	0.0	0.0
Appr. 2	SP-U	0.342	0.288	-0.041	0.111	0.009	0.0	0.0	0.684	0.051	0.626
Appr. 3	TS	0.685	0.593	0.347	0.377	0.425	0.0	0.0	0.0	0.0	0.0
Appr. 3	SP-TS	0.696	0.602	0.374	0.387	0.443	0.0	0.0	0.0	0.0	0.0
Appr. 3	SP-1	0.383	0.29	0.164	0.154	0.192	0.0	0.0	0.0	0.0	0.0
Appr. 3	SP-U	0.333	0.276	-0.045	0.063	0.003	0.005	0.0	0.685	0.35	0.637

Table 3: Avg. % Entropy Decrease and Entropy Difference for each approach in the Bayesian setting on `SVHN`.

model, we obtain $L = 200$ dropout predictions. We show the results in Figure 8; for results that include the base model, see Section D.2 of the Appendix. Overall, we can see that the results are similar to the point prediction setting, and the conclusions from Section 3.4 remain the same. We also see that the metrics are superior to the metrics in the point prediction setting, indicating that the aggregation of multiple predictions in a Bayesian model is beneficial in general.

In terms of ECE, using the softplus family of functions still proves superior to temperature scaling, although we observe that the difference is not as great as in the point prediction setting. For example, in Approach 2, all the methods perform similarly in ResNet, SE-Net, and DenseNet, although SP-TS still matches or performs better than TS in each model. However, there still are many situations where the ECE of TS can be improved significantly, such as in Xception. In terms of AUC, the results are similar to the point prediction setting, with TS and SP-TS performing the best, SP-TS having equal performance to TS, and SP-1 performing better than SP-U. Overall, the results still indicate that SP-TS strikes the best balance in terms of calibration and OOD detection performance. In Section D.3 of the Appendix, we compare each of the different Bayesian recalibration approaches. For SP-TS, we find that they perform similarly, but Approach 3 has the highest AUC, followed by Approach 2, and then Approach 1.

Impact of Homogenization. In Table 3, we show the Avg. % Entropy Decrease and Entropy Difference metrics for each of the approaches on `SVHN`, see Section D.1 for results on `LSUN`. The results are similar to those of the point prediction setting: only SP-U sees a decrease in entropy on some observations, and the ordering in entropy difference goes SP-TS ¿ TS ¿ SP-1 ¿ SP-U (notably, the average entropy difference for SP-U is now negative on Xception). We note that in Proposition 5, we only showed that homogenization guarantees that the entropy increases in the point prediction setting; this does not necessarily hold in the Bayesian setting. See Section B.2 of the Appendix for an example where this occurs. Nevertheless, the fact that we never see such an entropy decrease in the homogenizing

mappings, while we do in the non-homogenizing mapping, suggests that using homogenizing point-prediction mappings in the Bayesian setting is advisable if increasing the uncertainty of the predictions is desired. In Approach 1, we guarantee that the entropy of the individual dropout predictions increase, and in Approach 2, we guarantee that the output probability corresponding to the mean of the logit vectors increases in entropy. These properties may help explain why the actual output probability also increases in entropy, as well as why the results for Approaches 1 and 2 are similar.

Impact on Underconfidence. As in the point prediction setting, the results in Figure 8 do not present evidence of problematic underconfidence in OOD detection with SP-TS, as it still matches the performance of TS despite the fact that it increases the uncertainty in predictions more than TS. In Appendix Section D.5, we also show calibration underconfidence/overconfidence metrics for each method. As before, we find that SP-TS does result in increased underconfidence, which should be taken into consideration when selecting a recalibration method. The value of TS also increases compared to the point prediction setting, since it displays a lower level of overconfidence. SP-1 still offers a balance, with lower overconfidence than TS and lower underconfidence than SP-TS.

5. Conclusion

Based on the homogenization property of temperature scaling, we proposed an approach to recalibration based on learning a homogenizing softplus recalibration mapping by constraining the slope. We then presented experimental results showing the impact of the homogenization property in both the point prediction and Bayesian setting. We conclude that enforcing homogenization can result in significant improvements to OOD detection performance, while still maintaining better calibration over standard temperature scaling. Future work should investigate the impact of homogenization for more complicated recalibration functions. In the Bayesian setting, the effect that recalibration has on applications using the intermediate logit distributions, such as epistemic uncertainty estimation, should be investigated.

Acknowledgments

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This written work is authored by an employee of NTESS. The employee, not NTESS, owns the right, title and interest in and to the written work and is responsible for its contents. Any subjective views or opinions that might be expressed in the written work do not necessarily represent the views of the U.S. Government. The publisher acknowledges that the U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this written work or allow others to do so, for U.S. Government purposes. The

DOE will provide public access to results of federally sponsored research in accordance with the DOE Public Access Plan.

This work made use of the Illinois Campus Cluster, a computing resource that is operated by the Illinois Campus Cluster Program (ICCP) in conjunction with the National Center for Supercomputing Applications (NCSA) and which is supported by funds from the University of Illinois at Urbana-Champaign. This work also utilizes resources supported by the National Science Foundation's Major Research Instrumentation program, grant #1725729, as well as the University of Illinois at Urbana-Champaign. This work was supported by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists, Office of Science Graduate Student Research (SCGSR) program. The SCGSR program is administered by the Oak Ridge Institute for Science and Education for the DOE under contract number DE-SC0014664.

Appendix A. Proofs

A.1 Lemma for Proof of Proposition 2

In our proof of Proposition 2, we first introduce the following lemma, which states that if we take probability mass from a component with high probability and transfer it to a component with low probability, the entropy will increase.

Lemma 6 *Given a probability vector $\boldsymbol{p} \in \Delta_C$, let i, j be indices such that $p_i \leqslant p_j$. Then given $a \in [0, \frac{p_j - p_i}{2}]$, define*

$$\tilde{\boldsymbol{p}} = \boldsymbol{p} + a\boldsymbol{e}_i - a\boldsymbol{e}_j$$

where given $c \in \{1, \ldots, C\}$, we define $\boldsymbol{e}_c \in \mathbb{R}^C$ by

$$e_c = 1, \quad e_{c'} = 0 \text{ for } c' \neq c.$$

Then $\mathbb{H}(\tilde{\boldsymbol{p}}) \geqslant \mathbb{H}(\boldsymbol{p})$.

Proof Given $\boldsymbol{p} \in \Delta_C$, consider the function $g : \mathbb{R} \to \mathbb{R}$ defined by

$$\begin{aligned} g(a) &= \mathbb{H}(\boldsymbol{p} + a\boldsymbol{e}_i - a\boldsymbol{e}_j) \\ &= -(p_i + a)\log(p_i + a) - (p_j - a)\log(p_j - a) - \sum_{c=1}^{C} \mathbb{1}(c \neq i, c \neq j) p_c \log p_c. \end{aligned}$$

Taking the derivative with respect to a, we have that

$$g'(a) = (1 + \log(p_j - a)) - (1 + \log(p_i + a)) = \log\left(\frac{p_j - a}{p_i + a}\right).$$

Observe that $g'(\frac{p_j - p_i}{2}) = 0$ and $g'(a) > 0$ for $a \in (0, \frac{p_j - p_i}{2})$, so $g(a)$ is increasing on $[0, \frac{p_j - p_i}{2}]$. This means $\mathbb{H}(\tilde{\boldsymbol{p}}) = \mathbb{H}(\boldsymbol{p} + a\boldsymbol{e}_i - a\boldsymbol{e}_j) = g(a) \geqslant g(0) = \mathbb{H}(\boldsymbol{p})$. ■

A.2 Proof of Proposition 2

Proof First, we note that the fact that $\text{vratio}(\tilde{\boldsymbol{p}}) \geqslant \text{vratio}(\boldsymbol{p})$ immediately follows from the fact that if $c' = \arg\max_{c \in 1:C} \boldsymbol{p}$, then $p_{c'} \geqslant p_k$, so we must have that $p_{c'} \geqslant \tilde{p}_c$, i.e., the maximum probability decreases, so the variation ratio increases.

To show that the entropy increases, we first enumerate the elements of S_1, S_2 by $S_1 = \{u_1, \ldots, u_I\}$, $S_2 = \{v_1, \ldots, v_J\}$ and we define $a_i = \tilde{p}_{u_i} - p_{u_i}$ for $i \in \{1, \ldots, I\}$ and $b_j = p_{v_j} - \tilde{p}_{v_j}$ for $j \in \{1, \ldots, J\}$. By Condition 1, $\boldsymbol{a} \in \mathbb{R}^C, \boldsymbol{b} \in \mathbb{R}^J$ are non-negative. Note that we have

$$\tilde{\boldsymbol{p}} = \boldsymbol{p} + \sum_{i=1}^{I} \boldsymbol{e}_{u_i} a_i - \sum_{j=1}^{J} \boldsymbol{e}_{v_j} b_j.$$

Furthermore, note that

$$\begin{aligned}
\sum_{i=1}^{I} a_i - \sum_{j=1}^{J} b_j &= \sum_{i=1}^{I} \tilde{p}_{u_i} - p_{u_i} - \left(\sum_{j=1}^{J} p_{v_j} - \tilde{p}_{v_j} \right) \\
&= \sum_{i=1}^{I} \tilde{p}_{u_i} + \sum_{j=1}^{J} \tilde{p}_{v_j} - \sum_{i=1}^{I} p_{u_i} - \sum_{j=1}^{J} p_{v_j} \\
&= \sum_{c=1}^{C} \tilde{p}_c - \sum_{c=1}^{C} p_c \\
&= 0,
\end{aligned}$$

so let us define the common sum by $s = \sum_{i=1}^{I} a_i = \sum_{j=1}^{J} b_j$.

Next, define matrix $D \in \mathbb{R}^{J \times I}$ by $D_{ji} = b_j a_i / s$ and define $\boldsymbol{q}^{(0)} = \boldsymbol{p}$. For $j \in \{1, \ldots, J\}$, define

$$\begin{aligned}
\boldsymbol{q}^{(j)} &= \boldsymbol{q}^{(j,I)}, \quad \boldsymbol{q}^{(j,0)} = \boldsymbol{q}^{(j-1)}, \\
\boldsymbol{q}^{(j,i)} &= \boldsymbol{q}^{(j,i-1)} + D_{ji}(\boldsymbol{e}_{u_i} - \boldsymbol{e}_{v_j}), \text{ for } i \in \{1, \ldots, I\}.
\end{aligned}$$

Now, observe that

$$\begin{aligned}
\boldsymbol{q}^{(J)} &= \boldsymbol{p} + \sum_{j=1}^{J} \left(\sum_{i=1}^{I} D_{ji}(\boldsymbol{e}_{u_i} - \boldsymbol{e}_{v_j}) \right) \qquad (9) \\
&= \boldsymbol{p} + \sum_{j=1}^{J} \left(\sum_{i=1}^{I} (b_j a_i / s) (\boldsymbol{e}_{u_i} - \boldsymbol{e}_{v_j}) \right) \\
&= \boldsymbol{p} + \sum_{j=1}^{J} \left(\sum_{i=1}^{I} (b_j a_i / s) \boldsymbol{e}_{u_i} \right) - \sum_{i=1}^{I} \left(\sum_{j=1}^{J} (b_j a_i / s) \boldsymbol{e}_{v_j} \right) \\
&= \boldsymbol{p} + \left(\sum_{j=1}^{J} b_j \right) (s)^{-1} \left(\sum_{i=1}^{I} a_i \boldsymbol{e}_{u_i} \right) - \sum_{i=1}^{I} (a_i / s) \left(\sum_{j=1}^{J} b_j \boldsymbol{e}_{v_j} \right) \\
&= \boldsymbol{p} + \sum_{i=1}^{I} \boldsymbol{e}_{u_i} a_i - \sum_{j=1}^{J} \boldsymbol{e}_{v_j} b_j.
\end{aligned}$$

Hence, if we can prove that for all $i \in \{1, \dots, I\}, j \in \{1, \dots, J\}$, we have $D_{ji} \in \left[0, \frac{q_{v_j}^{(j,i-1)} - q_{u_i}^{(j,i-1)}}{2}\right]$, we can apply Lemma 1.1 to each step in (9) to complete the proof. Since $\boldsymbol{a}, \boldsymbol{b}$ are non-negative, we just need to show that

$$D_{ji} \leqslant (q_{v_j}^{(j,i-1)} - q_{u_i}^{(j,i-1)})/2.$$

Given $i \in \{1, \dots, I\}, j \in \{1, \dots, J\}$, note that

$$\begin{aligned} q_{v_j}^{(j,i-1)} - q_{u_i}^{(j,i-1)} &= p_{v_j} - \sum_{i'=1}^{i-1} D_{ji'} - (p_{u_i} + \sum_{j'=1}^{j-1} D_{j'i}) \\ &= p_{v_j} - \sum_{i'=1}^{i-1} a_{i'} b_j / s - (p_{u_i} + \sum_{j'=1}^{j-1} a_i b_{j'} / s). \end{aligned} \tag{10}$$

Using the assumption that $\tilde{p}_{i'} \leqslant \tilde{p}_{j'}$ for all $i' \in S_1, j' \in S_2$ (Condition 2), we have

$$\begin{aligned} \tilde{p}_{v_j} \geqslant \tilde{p}_{u_i} &\iff p_{v_j} - b_j \geqslant p_{u_i} + a_i \\ &\iff p_{v_j} - \left(\sum_{i'=1}^{I} a_{i'} b_j / s\right) \geqslant p_{u_i} + \left(\sum_{j'=1}^{J} a_i b_{j'} / s\right) && (11) \\ &\iff p_{v_j} - \left(\sum_{i'=1}^{i} a_{i'} b_j / s + \sum_{i'=i+1}^{I} a_{i'} b_j / s\right) \geqslant p_{u_i} + \left(\sum_{j'=1}^{j} a_i b_{j'} / s + \sum_{j'=j+1}^{J} a_i b_{j'} / s\right) \\ &\implies p_{v_j} - \left(\sum_{i'=1}^{i} a_{i'} b_j / s\right) \geqslant p_{u_i} + \left(\sum_{j'=1}^{j} a_i b_{j'} / s\right) && (12) \\ &\iff p_{v_j} - \sum_{i'=1}^{i-1} a_{i'} b_j / s - a_i b_j / s - \left(p_{u_i} + \sum_{j'=1}^{j-1} a_i b_{j'} / s + a_i b_j / s\right) \geqslant 0 \\ &\iff p_{v_j} - \sum_{i'=1}^{i-1} a_{i'} b_j / s - \left(p_{u_i} + \sum_{j'=1}^{j-1} a_i b_{j'} / s\right) \geqslant a_i b_j / s + a_i b_j / s \\ &\iff q_{v_j}^{(j,i-1)} - q_{u_i}^{(j,i-1)} \geqslant a_i b_j / s + a_i b_j / s && (13) \\ &\iff D_{ji} \leqslant \left(q_{v_j}^{(j,i-1)} - q_{u_i}^{(j,i-1)}\right)/2 \end{aligned}$$

where (11) follows from the identity $\sum_{i'=1}^{I} a_i = \sum_{j'=1}^{J} b_j = s$, (12) follows because a_i, b_j are non-negative, and in (13), we substitute in (10).

Finally, we conclude the proof by formally applying induction. Specifically, we want to prove the following statement for $j \in \{0, 1, \dots, J\}$:

$$P(j) : \mathbb{H}(\boldsymbol{q}^{(j)}) \geqslant \mathbb{H}(\boldsymbol{p}).$$

Base Case:

By definition, we have $\boldsymbol{q}^{(0)} = \boldsymbol{p}$, so $\mathbb{H}(\boldsymbol{q}^{(0)}) \geqslant \mathbb{H}(\boldsymbol{p})$.

Inductive Step:

Assume that $P(j-1)$ holds for some $j \in \{1, \ldots, J\}$, so we have that $\mathbb{H}(\boldsymbol{q}^{(j-1)}) \geqslant \mathbb{H}(\boldsymbol{p})$. By definition, we have $\boldsymbol{q}^{(j)} = \boldsymbol{q}^{(j,I)}$. Now, we use induction to prove the following statement for $i \in \{1, \ldots, I\}$:

$$Q(i) : \mathbb{H}(\boldsymbol{q}^{(j,i)}) \geqslant \mathbb{H}(\boldsymbol{p}).$$

Base Case:

By definition, $\boldsymbol{q}^{(j,0)} = \boldsymbol{q}^{(j-1)}$, so $\mathbb{H}(\boldsymbol{q}^{(j,0)}) = \mathbb{H}(\boldsymbol{q}^{(j-1)}) \geqslant \mathbb{H}(\boldsymbol{p})$.

Inductive Step:

Assume that $Q(i-1)$ holds for some $i \in \{1, \ldots, I\}$, so we have that $\mathbb{H}(\boldsymbol{q}^{(j,i-1)}) \geqslant \mathbb{H}(\boldsymbol{p})$. By definition, we have that

$$\boldsymbol{q}^{(j,i)} = \boldsymbol{q}^{(j,i-1)} + D_{ji}(\boldsymbol{e}_{u_i} - \boldsymbol{e}_{v_j})$$

and we have previously shown that $D_{ji} \in \left[0, \frac{q_{v_j}^{(j,i-1)} - q_{u_i}^{(j,i-1)}}{2}\right]$. Thus, we can apply Lemma 6 corresponding to indices u_i and v_j to conclude that $\mathbb{H}(\boldsymbol{q}^{(j,i)}) \geqslant \mathbb{H}(\boldsymbol{q}^{(j,i-1)}) \geqslant \mathbb{H}(\boldsymbol{p})$.

Thus, by induction, we have that $\mathbb{H}(\boldsymbol{q}^{(j,i)}) \geqslant \mathbb{H}(\boldsymbol{p})$ for all $i \in \{1, \ldots, I\}$, so we have that $\mathbb{H}(\boldsymbol{q}^{(j)}) = \mathbb{H}(\boldsymbol{q}^{(j,I)}) \geqslant \mathbb{H}(\boldsymbol{p})$.

Finally, by induction again, we have that $\mathbb{H}(\boldsymbol{q}^{(j)}) \geqslant \mathbb{H}(\boldsymbol{p})$ for all $j \in \{1, \ldots, J\}$, so we can conclude $\mathbb{H}(\tilde{\boldsymbol{p}}) = \mathbb{H}(\boldsymbol{q}^{(J)}) \geqslant \mathbb{H}(\boldsymbol{p})$. ■

A.3 Proof of Proposition 5

Proof Given $\boldsymbol{z} \in \mathbb{R}^C$, let $\tilde{\boldsymbol{p}}^{(1)} = (\boldsymbol{\eta} \circ \boldsymbol{g}^{(1)})(\boldsymbol{z})$, $\tilde{\boldsymbol{p}}^{(2)} = (\boldsymbol{\eta} \circ \boldsymbol{g}^{(2)})(\boldsymbol{z})$, and we define $\mathcal{I}$ to be the set of indices for which the recalibrated probability produced by $\boldsymbol{g}^{(1)}$ is greater than the recalibrated probability produced by $\boldsymbol{g}^{(2)}$:

$$\mathcal{I} = \left\{ c \in \{1, \ldots, C\} \;\middle|\; \tilde{p}_c^{(1)} \geqslant \tilde{p}_c^{(2)} \right\}.$$

Observe that $\mathcal{I}$ must be non-empty: suppose, by way of contradiction, that $\mathcal{I}$ is empty, so we have that $\tilde{p}_c^{(1)} < \tilde{p}_c^{(2)}$ for all $c \in \{1, \ldots, C\}$. Then, we also have that $\sum_{c=1}^C \tilde{p}_c^{(1)} < \sum_{c=1}^C \tilde{p}_c^{(2)}$. This is a contradiction, since $\tilde{\boldsymbol{p}}^{(1)}$ and $\tilde{\boldsymbol{p}}^{(2)}$ must sum to one, since they are both outputs of the softmax function.

Next, we define $k^* = \arg\max_{c \in \mathcal{I}} \tilde{p}_c^{(2)}$ (if there is more than one maximum, we let k^* be the largest numbered index). That is, k^* is an index in $\mathcal{I}$ that satisfies $\tilde{p}_{k^*}^{(2)} \geqslant \tilde{p}_c^{(2)}$ for all $c \in \mathcal{I}$. The goal is to show that k^* satisfies the role of "k" in Definition 1, where $\tilde{\boldsymbol{p}}^{(1)}$ plays the role of "$\tilde{\boldsymbol{p}}$" and $\tilde{\boldsymbol{p}}^{(2)}$ plays the role of "$\boldsymbol{p}$". That is, we want to show that the subsets S_1, S_2 given by

$$S_1 = \left\{ c \in \{1, \ldots, C\} \mid \tilde{p}_c^{(2)} \leqslant \tilde{p}_{k^*}^{(2)} \right\}, \quad S_2 = \left\{ c \in \{1, \ldots, C\} \mid \tilde{p}_c^{(2)} > \tilde{p}_{k^*}^{(2)} \right\}$$

satisfy Conditions 1 and 2 of Definition 1.

First, we show Condition 2, that for all $c \in S_1$ and $c' \in S_2$, we have that $\tilde{p}_c^{(1)} \leqslant \tilde{p}_{c'}^{(1)}$. Given $c \in S_1, c' \in S_2$, by definition, we have that $\tilde{p}_c^{(2)} \leqslant \tilde{p}_{k*}^{(2)}$ and $\tilde{p}_{c'}^{(2)} > \tilde{p}_{k*}^{(2)}$, so $\tilde{p}_c^{(2)} < \tilde{p}_{c'}^{(2)}$. Furthermore, since $\boldsymbol{g}^{(1)}, \boldsymbol{g}^{(2)}$ are order preserving, we also have that

$$\tilde{p}_c^{(2)} < \tilde{p}_{c'}^{(2)} \iff z_c < z_{c'} \iff \tilde{p}_c^{(1)} < \tilde{p}_{c'}^{(1)}.$$

Thus, Condition 2 of Definition 1 holds.

Next, we need to show that Condition 1 holds: that for all $c \in S_1$, $\tilde{p}_c^{(1)} \geqslant \tilde{p}_c^{(2)}$, and for all $c \in S_2, \tilde{p}_c^{(1)} < \tilde{p}_c^{(2)}$. For the second part, given $c \in S_2$, so that $\tilde{p}_c^{(2)} > \tilde{p}_{k*}^{(2)}$, we have by definition that $\tilde{p}_{k*}^{(2)} \geqslant \tilde{p}_{c'}^{(2)}$ for all $c' \in \mathcal{I}$, so we must have that $c \notin \mathcal{I}$, which by definition means $\tilde{p}_c^{(1)} < \tilde{p}_c^{(2)}$.

Finally, we show the first part of Condition 1. First, given $c \in \{1, \ldots, C\}$, we have that

$$\begin{aligned}
\tilde{p}_c^{(1)} \geqslant \tilde{p}_c^{(2)} &\iff \frac{\exp(g_c^{(1)}(\boldsymbol{z}))}{\sum_{c'=1}^{C} \exp(g_{c'}^{(1)}(\boldsymbol{z}))} \geqslant \frac{\exp(g_c^{(2)}(\boldsymbol{z}))}{\sum_{c'=1}^{C} \exp(g_{c'}^{(2)}(\boldsymbol{z}))} \\
&\iff d \cdot \exp(g_c^{(1)}(\boldsymbol{z})) \geqslant \exp(g_c^{(2)}(\boldsymbol{z})) \\
&\iff g_c^{(1)}(\boldsymbol{z}) - g_c^{(2)}(\boldsymbol{z}) + \log d \geqslant 0 \qquad (14)
\end{aligned}$$

where $d = \sum_{c'=1}^{C} \exp(g_{c'}^{(2)}(\boldsymbol{z})) / \sum_{c'=1}^{C} \exp(g_{c'}^{(1)}(\boldsymbol{z}))$. Now, given $c \in S_1$, so that $\tilde{p}_c^{(2)} \leqslant \tilde{p}_{k*}^{(2)}$, by the definition of the softmax function, we have that $g_c^{(2)}(\boldsymbol{z}) \leqslant g_{k*}^{(2)}(\boldsymbol{z})$. Since $\boldsymbol{g}^{(1)}$ is more homogenizing than $\boldsymbol{g}^{(2)}$, we have

$$|g_{k*}^{(1)}(\boldsymbol{z}) - g_c^{(1)}(\boldsymbol{z})| \leqslant |g_{k*}^{(2)}(\boldsymbol{z}) - g_c^{(2)}(\boldsymbol{z})| = g_{k*}^{(2)}(\boldsymbol{z}) - g_c^{(2)}(\boldsymbol{z}).$$

In particular, this implies the following:

$$g_{k*}^{(1)}(\boldsymbol{z}) - g_c^{(1)}(\boldsymbol{z}) \leqslant g_{k*}^{(2)}(\boldsymbol{z}) - g_c^{(2)}(\boldsymbol{z}) \iff g_c^{(1)}(\boldsymbol{z}) - g_c^{(2)}(\boldsymbol{z}) \geqslant g_{k*}^{(1)}(\boldsymbol{z}) - g_{k*}^{(2)}(\boldsymbol{z}). \qquad (15)$$

Since $k^* \in \mathcal{I}$, we have that $\tilde{p}_{k*}^{(1)} \geqslant \tilde{p}_{k*}^{(2)}$, so applying (14), we have

$$g_{k*}^{(1)}(\boldsymbol{z}) - g_{k*}^{(2)}(\boldsymbol{z}) + \log d \geqslant 0 \qquad (16)$$

and combining (15) and (16), we have

$$g_c^{(1)}(\boldsymbol{z}) - g_c^{(2)}(\boldsymbol{z}) + \log d \geqslant 0.$$

Finally, applying (14) again, this implies that $\tilde{p}_c^{(1)} \geqslant \tilde{p}_c^{(2)}$. Thus, Condition 1 of Definition 1 holds. This completes the proof. ■

Appendix B. Additional Numerical Examples

B.1 Examples of Entropy Increase and Variation Ratio Increase without Homogenization

In this section, we provide some examples of entropy increase without homogenization or variation ratio increase and variation ratio increase without homogenization or entropy increase. Consider the following probability vectors:

$$\boldsymbol{p} = \begin{bmatrix} 0.01 & 0.15 & 0.25 & 0.59 \end{bmatrix}, \quad \tilde{\boldsymbol{p}} = \begin{bmatrix} 0.01 & 0.01 & 0.4 & 0.58 \end{bmatrix}.$$

Here, $\tilde{\boldsymbol{p}}$ is not more homogenous than $\boldsymbol{p}$, because there does not exist a cutoff that divides the components into a group that increases in probability and decreases in probability. The variation ratio increases from 0.41 to 0.42 and the entropy decreases from 0.9885 to 0.7746. Next, consider the following probability vectors:

$$\boldsymbol{p} = \begin{bmatrix} 0.01 & 0.01 & 0.4 & 0.58 \end{bmatrix}, \quad \tilde{\boldsymbol{p}} = \begin{bmatrix} 0.001 & 0.2 & 0.218 & 0.581 \end{bmatrix}.$$

Again, $\tilde{\boldsymbol{p}}$ is not more homogenous than $\boldsymbol{p}$. In this case, the variation decreases from 0.42 to 0.419 and the entropy increases from 0.7746 to 0.9764.

B.2 Examples of Entropy Decrease in Bayesian Setting with Homogenizing Point Prediction Recalibration Mapping

In Section 4.2, we showed how to apply a point-prediction recalibration mapping to the Bayesian setting. In Section 3.2, we proved that in the point prediction setting, applying a homogenizing recalibration mapping to a logit prediction always increases the uncertainty in terms of homogenization. However, the same is not necessarily the case in the Bayesian setting. In this section, we provide an example. Consider a three-class setting where the CDF prediction $F : \mathbb{R}^3 \to [0, 1]$ is given by

$$F(\boldsymbol{z}) = \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3.98^2 \end{bmatrix} \right).$$

We can numerically verify that the corresponding probability prediction is nearly uniform:

$$\boldsymbol{p} = \mathbb{E}_{\boldsymbol{z} \sim F}[\boldsymbol{\eta}(\boldsymbol{z})] = [0.333, 0.333, 0.334].$$

Consider a temperature scaling recalibration mapping with $T = 0.5$. With temperature scaling, the corresponding recalibrated probability predictions are given by

$$\tilde{\boldsymbol{p}} = \mathbb{E}_{\boldsymbol{z} \sim F}[\boldsymbol{\eta}(0.5 \cdot \boldsymbol{z})] = [0.341, 0.341, 0.319] \qquad \text{(Approach 1)}$$
$$\tilde{\boldsymbol{p}} = \mathbb{E}_{\boldsymbol{z} \sim F}[\boldsymbol{\eta}(\boldsymbol{z} + (0, 0, 0.5)] = [0.312, 0.312, 0.376] \qquad \text{(Approach 2)}$$

which corresponds to a change in entropy from about 1.0986 to 1.0981 and 1.0947 for Approaches 1 and 2, respectively.

```
class BottleNeck(nn.Module):
expansion = 4
def __init__(self, in_channels, out_channels, stride=1):
    super().__init__()
    self.residual_function = nn.Sequential(
        nn.Conv2d(in_channels, out_channels, kernel_size=1, bias=False),
        nn.BatchNorm2d(out_channels),
        nn.ReLU(inplace=True),
        nn.Conv2d(out_channels, out_channels, stride=stride, kernel_size=3,
            padding=1, bias=False),
        nn.BatchNorm2d(out_channels),
        nn.ReLU(inplace=True),
        #nn.Dropout(),
        nn.Conv2d(out_channels, out_channels * BottleNeck.expansion,
            kernel_size=1, bias=False),
        nn.BatchNorm2d(out_channels * BottleNeck.expansion),
    )
```

Figure 9: Modification to `resnet.py`

Appendix C. Implementation Details

C.1 Training the Base Models

To train each model, we use the PyTorch implementation from `https://github.com/weiaicunzai/pytorch-cifar100`. Note that by default, no dropout layers are present in ResNet, DenseNet, SE-Net, and Xception, so we add in a dropout layer with dropout probability 0.5 within the model. In Figure 9, Figure 10, Figure 11, and Figure 12, we show code chunks that we added the layer into. We train each model for 150 epochs using the default parameters from the implementation: SGD optimizer with learning rate 0.1, momentum 0.9, and weight decay 5e-4. During training, a random crop, horizontal flip, and rotation is applied to the image. In addition, when training and testing, the input image is normalized by subtracting/dividing by the following mean and standard deviation:

$$\texttt{mean} = \begin{bmatrix}\texttt{0.5070751592371323} & \texttt{0.48654887331495095} & \texttt{0.4409178433670343}\end{bmatrix}$$
$$\texttt{std} = \begin{bmatrix}\texttt{0.2673342858792401} & \texttt{0.2564384629170883} & \texttt{0.27615047132568404}\end{bmatrix}$$

C.2 Obtaining Dropout Predictions

After training each model, we use it to obtain predictions on the `CIFAR-100`, `SVHN`, and `LSUN` data sets. For each data set, we include 10,000 observations, and we obtain 200 dropout predictions for each observation. During testing, we apply the same normalization as during training, but we do not apply the random crops, flips, and rotations. For `CIFAR-100`, we obtain predictions on all observations from the test split. For `SVHN`, we obtain predictions on a subset of the train split. For `LSUN`, we obtain predictions on a subset of the validation

```
class BottleneckResidualSEBlock(nn.Module):

expansion = 4

def __init__(self, in_channels, out_channels, stride, r=16):
    super().__init__()

    self.residual = nn.Sequential(
        nn.Conv2d(in_channels, out_channels, 1),
        nn.BatchNorm2d(out_channels),
        nn.ReLU(inplace=True),

        nn.Conv2d(out_channels, out_channels, 3, stride=stride, padding=1),
        nn.BatchNorm2d(out_channels),
        nn.ReLU(inplace=True),

        #nn.Dropout(),
        nn.Conv2d(out_channels, out_channels * self.expansion, 1),
        nn.BatchNorm2d(out_channels * self.expansion),
        nn.ReLU(inplace=True)
    )
```

Figure 10: Modification to `senet.py`

split. Also, we rescaled the `LSUN` images to be 32×32, the same dimensions as the `CIFAR-100` and `SVHN` images.

C.3 Recalibration

C.3.1 Alternative Parameterization

In our implementation of the recalibration methods, we use a slightly different parameterization for the softplus functions as the one in the presentation. Instead of learning three parameters a, b, c using a function of the form

$$g(x) = -(c/a)\log(1 + \exp(-a(x - b))),$$

we learn four parameters of a, b, c, u using a function of the form

$$g(x) = -(c/a)\log(u + \exp(-a(x - b))).$$

```
class ExitFLow(nn.Module):

def __init__(self):
    super().__init__()
    self.residual = nn.Sequential(
        nn.ReLU(),
        SeperableConv2d(728, 728, 3, padding=1),
        nn.BatchNorm2d(728),
        nn.ReLU(),
        SeperableConv2d(728, 1024, 3, padding=1),
        nn.BatchNorm2d(1024),
        nn.MaxPool2d(3, stride=2, padding=1)
    )

    self.shortcut = nn.Sequential(
        nn.Conv2d(728, 1024, 1, stride=2),
        nn.BatchNorm2d(1024)
    )

    self.conv = nn.Sequential(
        SeperableConv2d(1024, 1536, 3, padding=1),
        nn.BatchNorm2d(1536),
        nn.ReLU(inplace=True),
        #nn.Dropout(),
        SeperableConv2d(1536, 2048, 3, padding=1),
        nn.BatchNorm2d(2048),
        nn.ReLU(inplace=True)
    )
```

Figure 11: Modification to `xception.py`

```
class Bottleneck(nn.Module):
def __init__(self, in_channels, growth_rate):
    super().__init__()
    inner_channel = 4 * growth_rate
    self.bottle_neck = nn.Sequential(
        nn.BatchNorm2d(in_channels),
        nn.ReLU(inplace=True),
        nn.Conv2d(in_channels, inner_channel, kernel_size=1, bias=False),
        nn.BatchNorm2d(inner_channel),
        nn.ReLU(inplace=True),
        #nn.Dropout(),
        nn.Conv2d(inner_channel, growth_rate, kernel_size=3, padding=1,
            bias=False)
    )
```

Figure 12: Modification to `densenet.py`

Note that the addition of u is redundant if it is greater than 0, since we can absorb it into the b parameter:

$$\begin{aligned}
g(x) &= -(c/a)\log(u + \exp(-a(x-b))) \\
&= -(c/a)\log(u(1 + (1/u)\exp(-a(x-b)))) \\
&= -(c/a)\log u - (c/a)\log(1 + (1/u)\exp(-a(x-b))) \\
&= -(c/a)\log u - (c/a)\log(1 + \exp(-\log(u))\exp(-a(x-b))) \\
&= -(c/a)\log u - (c/a)\log(1 + \exp(-a(x-b) - \log(u))) \\
&= -(c/a)\log u - (c/a)\log\Big(1 + \exp\Big(-a\Big(x - \frac{-\log u - ab}{a}\Big)\Big)\Big) \\
&\sim -(c/a)\log\Big(1 + \exp\Big(-a\Big(x - \frac{-\log u - ab}{a}\Big)\Big)\Big)
\end{aligned}$$

and since we end up applying the softmax function to $g(x)$, the constant $(c/a)\log u$ term can be ignored. In addition, we constrain the parameters a, c, u to be nonnegative by squaring them in our implementation. Even though u is redundant, we find that adding it in helps the optimization. For SP-1 and SP-TS, we set initial parameters `a = 1, b = 1, u = 0.1`. For SP-U, we set initial parameters `a = 0.135, b = 1, u = 0.1`. For the shift experiments of Section 3.4.2, we initialize the shift parameter to a vector of 0s.

C.3.2 Experimental Settings

In each experiment, we do not train new models or generate new dropout predictions. To obtain variation over different seeds, we randomly divide the size-10,000 test data set into a validation data set of size 8,000 and a test data set of size 2,000. In addition, we randomly sample 2000 observations from the OOD data set to compute the OOD detection metrics. For each method and base model, we use the precomputed dropout predictions on the validation data set to learn the optimal recalibration mapping by gradient descent using the Adam optimizer with learning rate 0.1. In our experiments, to conduct recalibration in the point prediction setting, we simply throw away all of the dropout predictions, except for the first one. To conduct recalibration in the Bayesian setting, we utilize all of the dropout predictions according to the approaches described in Section 4. For each type of experiment, we need to train for a different number of epochs to achieve convergence. The settings used are shown in Table 4. We repeat each experiment over 20 random seeds.

Appendix D. Additional Experimental Results

D.1 Avg. % Entropy Decrease and Entropy Difference Tables on `LSUN`

In Table 5, Table 6, and Table 7, we show the same Avg. % Entropy Decrease and Entropy Difference values as in Table 1, Table 2, and Table 3, respectively, in the main text, but now with `LSUN` as the OOD data set. Overall, the results are similar to those with the `SVHN` data set, and do not change our conclusions. In Table 5, we see that with SP-U on Xception and Inception, the average entropy actually decreases after applying recalibration, and a greater proportion of observations have a decreased entropy than with `SVHN`. However, we note that the average entropy difference also decreases for the homogenizing methods, and from Figure 5, there does not appear to be an increased difference in relative performance

Model	Setting	TS	SP-TS	SP-1	SP-U
ResNet	PointPred	200	300	800	1200
SE-Net	PointPred	200	300	800	1200
Xception	PointPred	200	300	600	1300
DenseNet	PointPred	200	300	800	1200
Inception	PointPred	200	300	800	1400
ResNet	Shift	200	600	800	1200
SE-Net	Shift	200	600	800	1200
Xception	Shift	200	300	600	1300
DenseNet	Shift	200	600	800	1200
Inception	Shift	200	600	800	1400
ResNet	Approach 1	250	300	1100	1000
SE-Net	Approach 1	250	300	1000	300
Xception	Approach 1	250	300	600	1300
DenseNet	Approach 1	200	300	1000	400
Inception	Approach 1	200	300	800	1300
ResNet	Approach 2	200	400	1000	1000
SE-Net	Approach 2	200	300	1000	400
Xception	Approach 2	200	300	800	1100
DenseNet	Approach 2	200	300	1000	300
Inception	Approach 2	250	300	900	1100
ResNet	Approach 3	200	400	900	1000
SE-Net	Approach 3	200	600	1000	400
Xception	Approach 3	250	400	900	1200
DenseNet	Approach 3	250	300	1200	1000
Inception	Approach 3	200	300	800	1200

Table 4: Epochs used for each experiment. "Shift" refers to the approach from Section 3.4.2

	Avg. Entropy Difference					Avg. % Entropy Decrease				
Method	Res	SE	Xcept	Dense	Incept	Res	SE	Xcept	Dense	Incept
TS	1.129	0.926	0.396	0.822	0.448	0.0	0.0	0.0	0.0	0.0
SP-TS	1.142	0.937	0.419	0.84	0.46	0.0	0.0	0.0	0.0	0.0
SP-1	0.664	0.492	0.18	0.441	0.193	0.0	0.0	0.0	0.0	0.0
SP-U	0.595	0.43	-0.029	0.344	-0.012	0.0	0.0	0.681	0.02	0.696

Table 5: Avg. % Entropy Decrease and Avg. Entropy Difference metrics on `LSUN`.

	Avg. Entropy Difference					Avg. % Entropy Decrease				
Method	Res	SE	Xcept	Dense	Incept	Res	SE	Xcept	Dense	Incept
SP-TS	1.141	0.936	0.421	0.843	0.461	0.0	0.0	0.0	0.0	0.0
SP-1	0.572	0.443	0.163	0.431	0.174	0.0	0.0	0.0	0.0	0.0
SP-U	0.424	0.319	-0.109	0.281	-0.1	0.09	0.146	0.717	0.25	0.714

Table 6: Avg. % Entropy Decrease and Entropy Difference when learning additional shift on `LSUN`.

with the homogenizing methods. The same observations also apply to Table 6 and Table 7, where the average entropy differences decreases on `LSUN` compared with `SVHN`, but there is not much difference in the difference in relative performance with other methods.

D.2 Evaluation Metrics for Base Model in Bayesian Setting

In Figure 13, we show the results from Figure 8, but now with the base (unrecalibrated) model added. Similar to the results from Figure 5, the base model suffers from poor calibration, although the degree of miscalibration is less than in the point prediction setting. In terms of OOD detection performance, the base model is also more competitive, now beating SP-U on Inception on all three approaches, and DenseNet on Approach 1. However, it still performs significantly worse than TS and SP-TS in all cases. Similar to how SP-TS being more homogenizing than TS helps maintain the OOD detection performance of TS, it appears that SP-1 being homogenizing (more homogenizing than the base model) helps prevent the OOD detection performance from decreasing.

D.3 Comparison of the Bayesian Approaches

We compare the evaluation metrics for each of the three Bayesian approaches. Which one should be used? We show the results in Figure 14. We observe that each method produces relatively similar results, but there are a few differences. For temperature scaling, we see that Approach 2 approach performs the best in terms of ECE, and always outperforms Approach 1. This aligns with the intuition that the complex transformation that Approach 1 produces may not be desirable. Approach 3 performs best in terms of OOD detection performance on TS and SP-TS, but Approach 2 performs better on SP-1 and SP-U. Approach 1 performs the worst overall, and never has the advantage in AUC in any experiment.

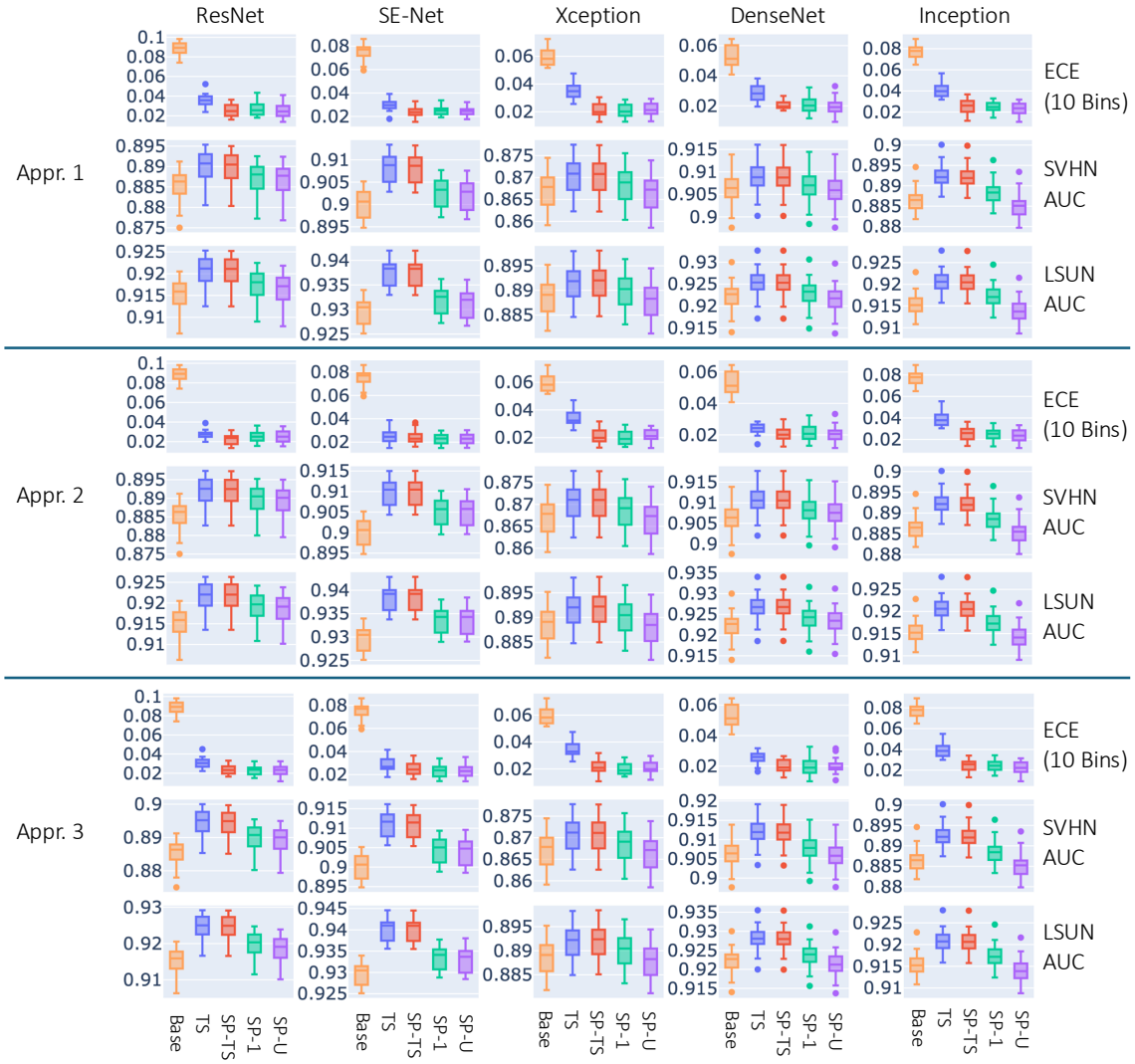

Figure 13: Evaluation metrics for each method, including and neural network model on `CIFAR-100` in the Bayesian setting.

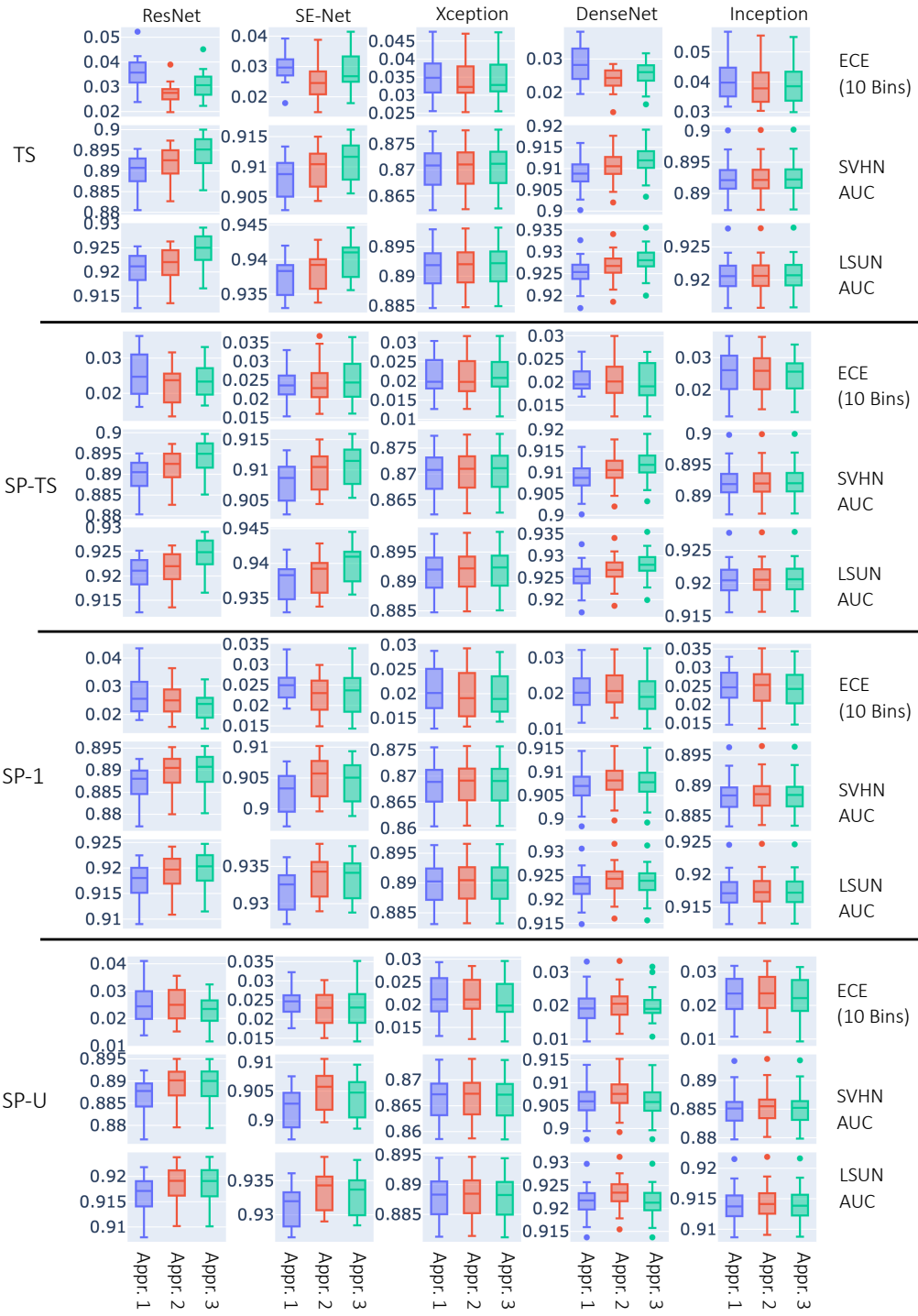

Figure 14: Comparison between the three different Bayesian recalibration approaches.

		Avg. Entropy Difference					Avg. % Entropy Decrease				
Type	Method	Res	SE	Xcept	Dense	Incept	Res	SE	Xcept	Dense	Incept
Appr. 1	TS	0.677	0.558	0.324	0.354	0.402	0.0	0.0	0.0	0.0	0.0
Appr. 1	SP-TS	0.688	0.568	0.347	0.367	0.415	0.0	0.0	0.0	0.0	0.0
Appr. 1	SP-1	0.337	0.236	0.141	0.138	0.167	0.0	0.0	0.0	0.0	0.0
Appr. 1	SP-U	0.253	0.209	-0.084	0.03	-0.045	0.011	0.0	0.764	0.453	0.753
Appr. 2	TS	0.64	0.535	0.327	0.352	0.399	0.0	0.0	0.0	0.0	0.0
Appr. 2	SP-TS	0.644	0.54	0.348	0.357	0.41	0.0	0.0	0.0	0.0	0.0
Appr. 2	SP-1	0.352	0.254	0.144	0.141	0.168	0.0	0.0	0.0	0.0	0.0
Appr. 2	SP-U	0.308	0.253	-0.076	0.093	-0.034	0.0	0.0	0.763	0.066	0.741
Appr. 3	TS	0.695	0.572	0.327	0.371	0.401	0.0	0.0	0.0	0.0	0.0
Appr. 3	SP-TS	0.702	0.577	0.351	0.379	0.413	0.0	0.0	0.0	0.0	0.0
Appr. 3	SP-1	0.36	0.256	0.145	0.14	0.166	0.0	0.0	0.0	0.0	0.0
Appr. 3	SP-U	0.294	0.24	-0.081	0.038	-0.043	0.001	0.0	0.763	0.46	0.752

Table 7: Avg. % Entropy Decrease and Entropy Difference for each approach in the Bayesian setting on `LSUN`.

D.4 Experimental Results Using Variation Ratio

We show the results when we use the variation ratio as the OOD score function rather than the entropy in Figure 15. First, we see that entropy results in significantly higher AUC across each set of experiments. Intuitively, we might expect that entropy would perform better at OOD detection, because it uses the entire probability vector, whereas variation ratio only uses the probability of the most confident class. Otherwise, we see that similar conclusions follow when using both methods. In particular, enforcing homogenization is still beneficial, as we still have that TS and SP-TS outperform SP-1, which outperforms SP-U. However, in contrast to the results using entropy, we see that SP-TS has slightly worse performance than TS. Regardless, this does not change the conclusion that SP-TS strikes the best balance between calibration and OOD detection performance, as the difference is small, and using entropy as the OOD score function is preferred anyway, where the performance is equal.

D.5 Analysis of Overconfidence and Underconfidence

In this section, we show additional evaluation metrics to assess the impact of recalibration on overconfidence and underconfidence. To do so, we utilize three related metrics to the ECE. For reference, given a test data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{T}$ and a probabilistic classifier $\boldsymbol{q} : \mathcal{X} \to \Delta_C$, the ECE with M bins is given by:

$$\text{ECE} = \frac{1}{T} \sum_{m=1}^{M} |B_m| |\text{conf}(B_m) - \text{acc}(B_m)| \tag{17}$$

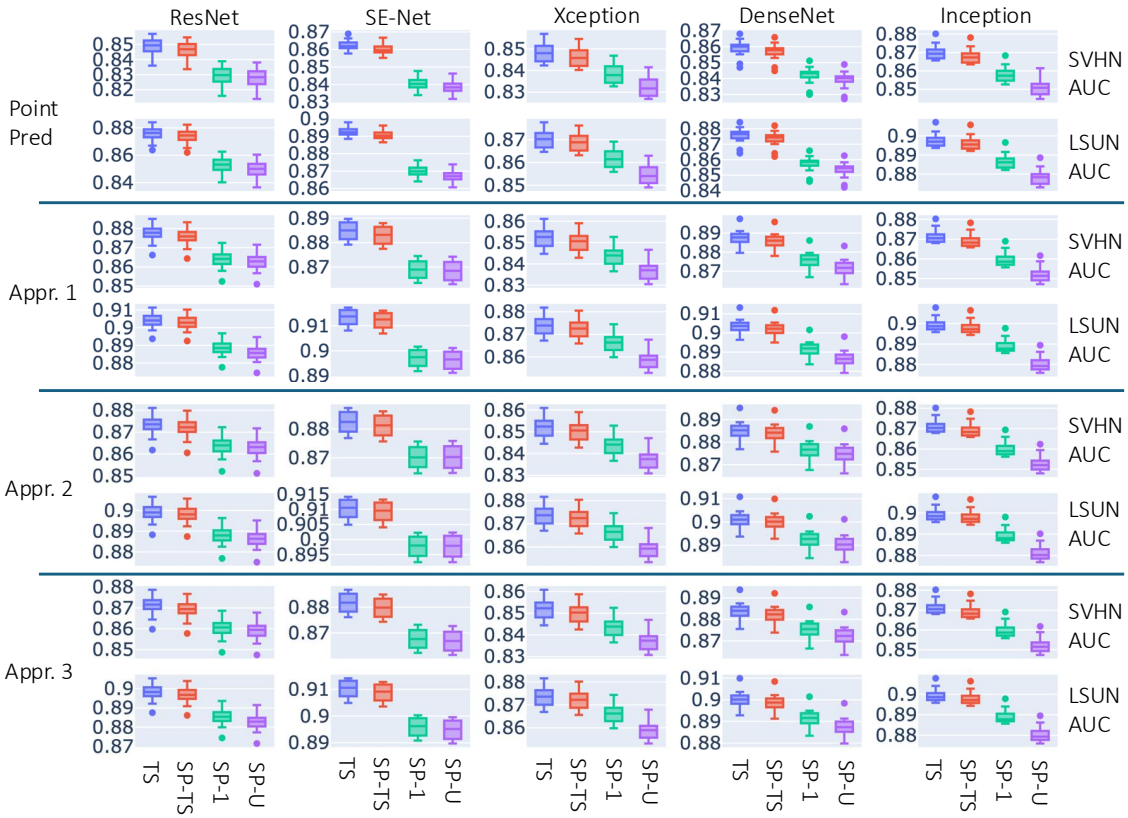

Figure 15: Evaluation metrics when using variation ratio as the OOD score function.

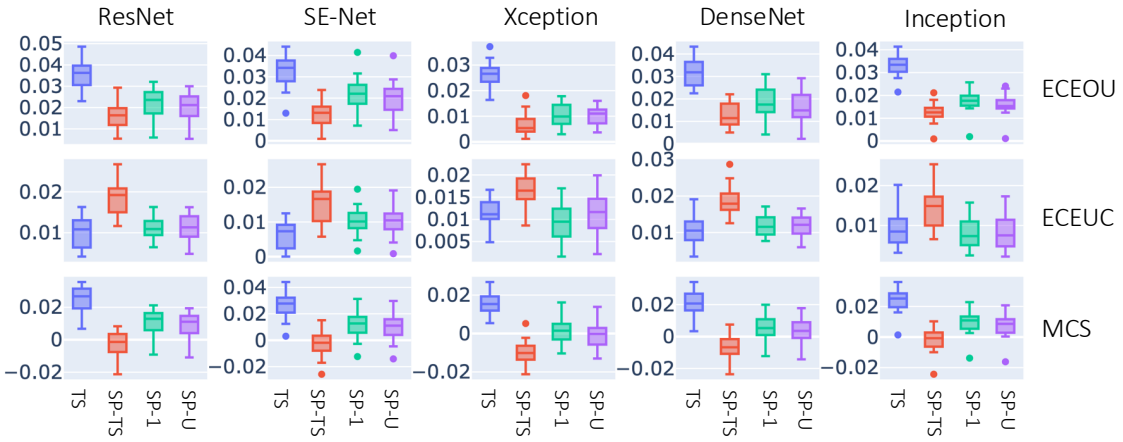

Figure 16: Boxplots of overconfidence/underconfidence metrics.

where for $m \in \{1, \dots, M\}$, the set of observations that belong to bin m is denoted by $B_m = \left\{ i \in \{1, \dots, T\} \mid \frac{m-1}{M} < \text{conf}(\boldsymbol{x}_i) \leqslant \frac{m}{M} \right\}$ (with the first bin also including 0) and

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \max_{c \in 1:C} q_c(\boldsymbol{x}_i), \quad \text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}\Big(\arg\max_{c \in 1:C} q_c(\boldsymbol{x}_i) = y_i \Big)$$

denote the average confidence and accuracy of each bin, respectively (note that the accuracy is computed with the classifier that outputs the highest-probability class as the prediction).

As discussed in Pearce and Meger (2022), the ECE can be decomposed into portions reflecting the overconfidence and underconfidence of the predictions by splitting the previous absolute value into positive and negative components, as follows:

$$\text{ECEUC} = \frac{1}{T} \sum_{m=1}^{M} |B_m| \max\big(\text{acc}(B_m) - \text{conf}(B_m), 0\big),$$
$$\text{ECEOC} = \frac{1}{T} \sum_{m=1}^{M} |B_m| \max\big(\text{conf}(B_m) - \text{acc}(B_m), 0\big).$$

This allows us to determine how much of the ECE is a result of overconfidence, and how much is a result of underconfidence. A related metric is known as the *miscalibration score* (MCS) (Ao et al., 2023), or the *net calibration error* (Groot and Valdenegro-Toro, 2024), which simply removes the absolute value bars from (17):

$$\text{MCS} = \frac{1}{T} \sum_{m=1}^{M} |B_m| \big(\text{conf}(B_m) - \text{acc}(B_m)\big)$$

The significance of the MCS is that it describes the direction of miscalibration in the ECE; a positive value signifies overconfidence, while a negative value signifies underconfidence.

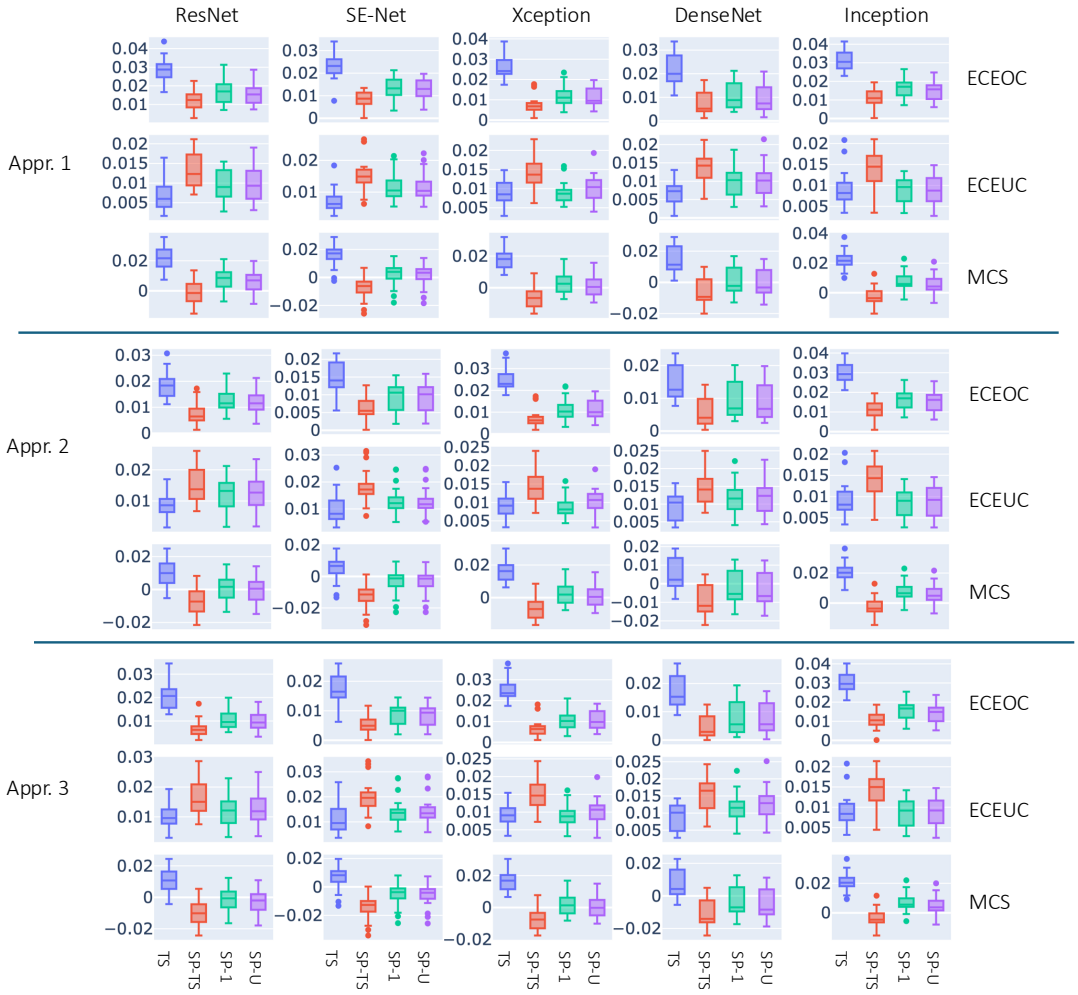

Figure 17: Boxplots of overconfidence/underconfidence metrics in the Bayesian setting.

D.5.1 Results in the Point Prediction Setting

We show boxplots for the overconfidence/underconfidence metrics for each method and model in the point prediction setting in Figure 16. For brevity, we omit results the base (unrecalibrated) model, since almost all of the ECE comes from overconfidence. First, we observe that TS, SP-1, and SP-U tend to skew towards overconfidence and SP-TS skews towards underconfidence. However, the degree of overconfidence in TS is greater than the other methods; as indicated by the lower MCS, SP-TS, SP-1, and SP-U are much more balanced in terms of overconfidence and underconfidence. If low underconfidence is especially valued, then the results slightly favor TS. However, SP-1 does nearly as well in underconfidence, while also having significantly lower overconfidence. If low overconfidence is valued, then SP-TS is the clear choice as it has the lowest overall ECEOU in for each model. If overconfidence and underconfidence are treated equally, then all of the softplus methods are reasonable choices.

D.5.2 Results in the Bayesian Setting

In Figure 17, we show the metrics with the three Bayesian approaches. As before, TS, SP-1, and SP-U skew towards overconfidence, while SP-TS skews towards underconfidence. The results from Approach 1 match the point prediction setting, with SP-TS, SP-1, and SP-U being more balanced in terms of overconfidence and underconfidence, compared to TS. Notably, the degree of overconfidence in TS is less than in the point prediction setting. This makes sense, since as we saw in Figure 8, the performance of TS is closer to the softplus recalibration methods overall in the Bayesian setting. In Approaches 2 and 3, we notice that SP-TS now skews towards underconfidence to a greater extent, having a lower MCS (although the ECEUC does not noticeably get higher). In addition, the overconfidence of TS is reduced further; this is most noticeable on ResNet, SE-Net, and DenseNet. As before, if low underconfidence is prioritized, then TS is the preferred choice, albeit now to a greater extent, and SP-1 still serves as an option with lower ECEOU than TS, but higher ECEUC. SP-TS now has an even lower overconfidence, especially in Approaches 2 and 3, and if overconfidence and underconfidence are treated equally, then all of the softplus methods are still reasonable choices.

References

Shuang Ao, Stefan Rueger, and Advaith Siddharthan. Two sides of miscalibration: Identifying over and under-confidence prediction for network calibration. In *Uncertainty in Artificial Intelligence*, pages 77–87. PMLR, 2023.

Sergio A Balanya, Juan Maroñas, and Daniel Ramos. Adaptive temperature scaling for robust calibration of deep neural networks. *Neural Computing and Applications*, pages 1–23, 2024.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern CCecognition*, pages 1251–1258, 2017.

Chi-Keung Chow. An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, (4):247–254, 1957.

Zhipeng Ding, Xu Han, Peirong Liu, and Marc Niethammer. Local temperature scaling for probability calibration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6889–6899, 2021.

Yasushi Esaki, Akihiro Nakamura, Keisuke Kawano, Ryoko Tokuhisa, and Takuro Kutsuna. Accuracy-preserving calibration via statistical modeling on probability simplex. In *International Conference on Artificial Intelligence and Statistics*, pages 1666–1674. PMLR, 2024.

Di Feng, Lars Rosenbaum, and Klaus Dietmayer. Towards safe autonomous driving: Capture uncertainty in the deep neural network for LIDAR 3D vehicle detection. In *International Conference on Intelligent Transportation Systems*, pages 3266–3273. IEEE, 2018.

Lior Frenkel and Jacob Goldberger. Network calibration by class-based temperature scaling. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 1486–1490. IEEE, 2021.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059. PMLR, 2016.

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

Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 56 (Suppl 1):1513–1589, 2023.

Tobias Groot and Matias Valdenegro-Toro. Overconfidence is key: Verbalized uncertainty evaluation in large language and vision-language models. In *Proceedings of the 4th Workshop on Trustworthy Natural Language Processing*, pages 145–171, Mexico City, Mexico, 2024. Association for Computational Linguistics. doi: 0.18653/v1/2024.trustnlp-1.13.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=Hkg4TI9xl`.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

Tom Joy, Francesco Pinto, Ser-Nam Lim, Philip HS Torr, and Puneet K Dokania. Sample-dependent adaptive temperature scaling for improved calibration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14919–14926, 2023.

Benjamin Kompa, Jasper Snoek, and Andrew L Beam. Second opinion needed: Communicating uncertainty in medical machine learning. *NPJ Digital Medicine*, 4(1):4, 2021.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Gerhard Krumpl, Henning Avenhaus, Horst Possegger, and Horst Bischof. ATS: Adaptive temperature scaling for enhancing out-of-distribution detection methods. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3864–3873, 2024.

Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with Dirichlet calibration. *Advances in Neural Information Processing Systems*, 32, 2019.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30, 2017.

Max-Heinrich Laves, Sontje Ihler, Jacob F Fast, Lüder A Kahrs, and Tobias Ortmaier. Well-calibrated regression uncertainty in medical imaging with deep learning. In *Medical Imaging with Deep Learning*, pages 393–412. PMLR, 2020a.

Max-Heinrich Laves, Sontje Ihler, Karl-Philipp Kortmann, and Tobias Ortmaier. Calibration of model uncertainty for dropout variational inference. *arXiv preprint arXiv:2006.11584*, 2020b.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, 33:21464–21475, 2020.

Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. *Advances in Neural Information Processing Systems*, 34:15682–15694, 2021.

Marcin Możejko, Mateusz Susik, and Rafał Karczewski. Inhibited softmax for uncertainty estimation in neural networks. *arXiv preprint arXiv:1810.01861*, 2018.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using Bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, volume 2011, page 7. Granada, Spain, 2011.

Jonathan Pearce and David Meger. Adaptive Confidence Calibration. *Proceedings of the Canadian Conference on Artificial Intelligence*, 2022.

John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3):61–74, 1999.

Amir Rahimi, Amirreza Shaban, Ching-An Cheng, Richard Hartley, and Byron Boots. Intra order-preserving functions for calibration of multi-class neural networks. *Advances in Neural Information Processing Systems*, 33:13456–13467, 2020.

Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. *Advances in Neural Information Processing Systems*, 32, 2019.

Silvia Seoni, Vicnesh Jahmunah, Massimo Salvi, Prabal Datta Barua, Filippo Molinari, and U Rajendra Acharya. Application of uncertainty quantification to artificial intelligence in healthcare: A review of last decade (2013–2023). *Computers in Biology and Medicine*, page 107441, 2023.

Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. In *International Conference on Machine Learning*, pages 20827–20840. PMLR, 2022.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, Inception-ResNet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.