Robust Calibration with Multi-domain Temperature Scaling

Yaodong Yu
University of California, Berkeley
yyu@eecs.berkeley.edu

Stephen Bates
University of California, Berkeley
stephenbates@cs.berkeley.edu

Yi Ma
University of California, Berkeley
yima@eecs.berkeley.edu

Michael I. Jordan
University of California, Berkeley
jordan@cs.berkeley.edu

Abstract

Uncertainty quantification is essential for the reliable deployment of machine learning models to high-stakes application domains. Uncertainty quantification is all the more challenging when training distribution and test distribution are different, even if the distribution shifts are mild. Despite the ubiquity of distribution shifts in real-world applications, existing uncertainty quantification approaches mainly study the in-distribution setting where the train and test distributions are the same. In this paper, we develop a systematic calibration model to handle distribution shifts by leveraging data from multiple domains. Our proposed method—multi-domain temperature scaling—uses the heterogeneity in the domains to improve calibration robustness under distribution shift. Through experiments on three benchmark data sets, we find our proposed method outperforms existing methods as measured on both in-distribution and out-of-distribution test sets.

1 Introduction

To make learning systems reliable and fault-tolerant, predictions must be accompanied by uncertainty estimates. A significant challenge to accurately codifying uncertainty is the distribution shift that typically arises over the course of a system’s deployment [Quiñonero-Candela et al., 2008]. For example, suppose health providers from 20 different hospitals employ a model to make diagnostic predictions from fMRI data. The distributions across hospitals could be quite different as a result of differing patient populations, machine conditions, and so on. In such a setting, it is critical to provide uncertainty quantification that is valid for every hospital—not just on average across all hospitals. Going even further, our uncertainty quantification should be informative when a new 21st hospital goes online, even if the distribution shifts from those already encountered. As another example, a centralized model is trained on training data from existing clients in federated learning. It is important for the central server to provide uncertainty quantification for every client. Similar to the fMRI example, the centralized model should still produce valid uncertainty quantification for unseen new clients. Another example is applying animal recognition models on images in wildlife monitoring, where one set of camera traps corresponds to one domain, and the model will be deployed under distribution shift, i.e., new camera traps. In this work, we study calibration in the multi-domain setting. We find that by requiring accurate calibration across all observed domains, our method provides more accurate uncertainty quantification on unseen domains.

Calibration is a core topic in learning [Platt et al., 1999; Naïemi et al., 2015; Gal and Ghahramani, 2016; Lakshminarayanan et al., 2017; Guo et al., 2017; Bates et al., 2021], but most techniques are targeted at settings with no distribution shift. To see this, we consider a simple experiment on the...
Figure 1: Reliability diagrams and expected calibration error histograms for temperature scaling with a ResNet-50 on ImageNet-C. We use temperature scaling to obtain adjusted confidences for the ResNet-50 model. (a) Reliability diagram evaluated on the pooled data of ImageNet-C. (b) Reliability diagram evaluated on data from one domain (Gaussian corruption with severity 5) in ImageNet-C. (c) Calibration evaluated on every domain in ImageNet-C as well as the pooled ImageNet-C (measured in ECE, lower is better).

ImageNet-C [Hendrycks and Dietterich, 2019] dataset, which consists of 76 domains. Here, each domain corresponds to one type of data corruption applied with a certain severity. We apply the temperature scaling technique [Guo et al., 2017] on the pooled data from all domains. In Figure 1(a) and 1(b), we display the reliability diagrams for the pooled data and for one individual domain. We find that even under a relatively mild distribution shift—i.e., subpopulation shift from the mixture of all domains to the single domain—temperature scaling does not produce calibrated confidence estimates on the stand-alone domain. This behavior is pervasive; in Figure 1(c), we see that the calibration on individual domains is much worse than the the reliability diagram from the pooled data would suggest.

To address this issue, we develop a new algorithm, multi-domain temperature scaling, that leverages multi-domain structure in the data. Our algorithm takes a base model and learns a calibration function that maps each input to a different temperature parameter that is used for adjusting confidence in the base model. Empirically, we find our algorithm significantly outperforms temperature scaling on three real-world multi-domain datasets. In particular, in contrast to temperature scaling, our proposed algorithm is able to provide well-calibrated confidence on each domain. Moreover, our algorithm largely improves robustness of calibration under distribution shifts. This is expected, because if the calibration method performs well on every domain, it is likely to have learned some structure that generalizes to unseen domains. Theoretically, we analyze the multi-domain calibration problem in the regression setting, providing guidance about the conditions under which robust calibration is possible.

**Contributions.** The main contributions of our work are as follows: Algorithmically, we develop a new calibration method that generalizes the widely used temperature scaling concept from single-domain to multi-domain. The proposed new method exploits multi-domain structure in the data distribution, which enables model calibration on every domain. We conduct detailed experiments on three real-world multi-domain datasets and demonstrate that our method significantly outperforms existing calibration methods on both in-distribution domains and unseen out-of-distribution domains. Theoretically, we study multi-domain calibration in the regression setting and develop a theoretical understanding of robust calibration in this setting.

**Related Work**

**Calibration methods.** There is a large literature on calibrating the well-trained machine learning models, including histogram binning [Zadrozny and Elkan, 2001], isotonic regression [Zadrozny and Elkan, 2002], conformal prediction Vovk et al. [2005], Platt scaling [Platt et al., 1999], and temperature scaling [Guo et al., 2017]. These calibration methods apply a validation set and post-process the model outputs. As shown in Guo et al. [2017], temperature scaling, a simple method that uses a single (temperature) parameter for rescaling the logits, performs surprisingly well on calibrating confidences for deep neural networks. We focus on this approach in our work. More broadly, there has been much recent work develop methods to improve calibration for deep learning models, including
augmentation-based training [Thulasidasan et al., 2019; Hendrycks et al., 2019b], calibration for neural machine translation [Kumar and Sarawagi, 2019], neural stochastic differential equation [Kong et al., 2020], self-supervised learning [Hendrycks et al., 2019a], ensembling [Lakshminarayanan et al., 2017], and Bayesian neural networks [Gal and Ghahramani, 2016; Gal et al., 2017], as well as statistical guarantees for calibration with black-box models Angelopoulos et al. [2021].

### Calibration under distribution shifts

Ovadia et al. [2019] conduct an empirical study on model calibration under distribution shifts and find that models are much less calibrated under distribution shifts. Minderer et al. [2021] revisit calibration of recent state-of-the-art image classification models under distribution shifts and study the relationship between calibration and accuracy. Wald et al. [2021] study model calibration and out-of-distribution generalization. Other works consider providing uncertainty estimates under structured distribution shifts, such as covariate shift [Tibshirani et al., 2019; Park et al., 2021], label shift [Podkopaev and Ramdas, 2021], and f-divergence balls [Cauchi et al., 2020]. Another line of work studies calibration in the domain adaptation setting [Wang et al., 2020; Park et al., 2020], which require unlabeled samples from the target domain.

### 2 Problem setup

#### Notation

We denote the input space and the label set by $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} = \{1, \ldots, J\}$. We let $[x]_i$ denote the $i$-th element of vector $x$. We use $\mathcal{P}(X)$ to denote the marginal feature distribution on input space $\mathcal{X}$, $\mathcal{P}(Y|X)$ to denote the conditional distribution, and $\mathcal{P}(X, Y)$ to denote the joint distribution. For the multiple domains scenario, we let $\mathcal{P}_k(X)$ and $\mathcal{P}_k(Y|X)$ denote the feature distribution and conditional distribution for the $k$-th domain. We let $f: \mathcal{X} \rightarrow \mathbb{R}^d$ denote the base model, e.g., a deep neural network, where $J$ is the total number of classes. We assume $f$ returns an (unnormalized) vector of logits. Throughout the paper, the base model is trained with training data and will not be modified.

The class prediction of model $f$ on input $x \in \mathcal{X}$ is denoted by $\hat{y} = \operatorname{argmax}_{j \in \{1, \ldots, J\}} [f(x; \theta)]_j$. We use $\mathbb{1}\{\cdot\}$ to represent the indicator function. We use $h(\cdot; f, \beta): \mathcal{X} \rightarrow [0, 1]$ to denote a **calibration map** parameterized by $\beta$ that takes an input $x \in \mathcal{X}$ and returns a confidence score—this is a post-processing of the base model $f$. We let $\hat{\pi} = h(x; f, \beta) \in [0, 1]$ denote the confidence estimate for sample $x$ when using model $f$. For instance, if we have 100 predictions $\{\hat{y}_1, \ldots, \hat{y}_{100}\}$ with confidence $\hat{\pi}_1 = \cdots = \hat{\pi}_{100} = 0.7$, then the accuracy of $f$ is expected to be $70\%$ on these 100 samples (if the confidence estimate is well calibrated). Data from the domains $\mathcal{P}_1, \ldots, \mathcal{P}_K$ are used for learning the calibration models, and we call the **in-distribution** (InD) domains. We use $\mathcal{P}$ to denote the unseen **out-of-distribution** (OOD) domain which is not used for calibrating the base model. Our goal is to learn a calibration map $h$ that is well calibrated on the OOD domain $\mathcal{P}$. To do this, we will learn a calibration map that does well on all InD domains simultaneously.

To measure calibration, we first review the definition of approximate expected calibration error.

#### Definition 2.1 (ECE)

For a set of samples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ with $(x_i, y_i) \overset{i.i.d.}\sim \mathcal{P}(X, Y)$, the (empirical) expected calibration error (ECE) with $M$ bins evaluated on $\mathcal{D}$ is defined as

$$
\text{ECE}(\mathcal{D}, M) = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{Acc}(B_m) - \text{Conf}(B_m)|,
$$

and $B_m$, $\text{acc}(B_m)$, $\text{conf}(B_m)$ are defined as

$B_m = \{i \in [n]: \hat{\pi}_i \in ((m-1)/M, m/M]\}$,

$\text{Acc}(B_m) = (1/|B_m|) \sum_{i \in B_m} \mathbb{1}\{\hat{y}_i = y_i\}$,

$\text{Conf}(B_m) = (1/|B_m|) \sum_{i \in B_m} \hat{\pi}_i$, $\hat{\pi}_i$ and $\hat{y}_i$ are the confidence and predicted label of sample $x_i$.

The empirical ECE defined in Eq. (1) approximates the expected calibration error (ECE) $\mathbb{E}[|p - \mathcal{P}(\hat{y} = y|\hat{\pi} = p)|$ with bin size equal to $M$ Naeini et al. [2015]; Guo et al. [2017]; see Lee et al. [2022] for statistical results about the empirical ECE as an estimator. The perfect calibrated map corresponds to the case when $\mathcal{P}(\hat{y} = y|\hat{\pi} = p) = p$ holds for all $p \in [0, 1]$.

#### Multi-domain calibration

Although the standard ECE measurement in Eq. (1) provides informative evaluations for various calibration methods in the single-domain scenario, it does not provide fine-grained evaluations when the dataset consists of multiple domains, $\mathcal{P}_1, \ldots, \mathcal{P}_K$. It is possible that the
We propose our algorithm—multi-domain temperature scaling—that aims to improve the calibration of the predictions of a model. At a high level, we propose to learn a calibration model that maps samples from the input space $X$ to the temperature space $\mathbb{R}_+$. To do this, we first compute feature embeddings of samples from different domains, and then we apply temperature scaling on each domain $k$ separately, then TS is able to produce calibrated confidence on every domain. Therefore, the question becomes how to “aggregate” these temperature scaling models and learn one calibration model, denoted by $\hat{h}$, that has similar performance to the $k$-th calibration model $h_k$ evaluated on domain $k$ for every $k \in [K]$.

At a high level, we propose to learn a calibration model that maps samples from the input space $X$ to the temperature space $\mathbb{R}_+$. To do this, we first compute feature embeddings of samples from different domains, and then we apply temperature scaling on each domain $k$ separately, then TS is able to produce calibrated confidence on every domain. Therefore, the question becomes how to “aggregate” these temperature scaling models and learn one calibration model, denoted by $\hat{h}$, that has similar performance to the $k$-th calibration model $h_k$ evaluated on domain $k$ for every $k \in [K]$.

To start with, we learn the temperature parameter $\hat{T}$ for each domain $k$ by solving the following convex optimization problem,

$$
\min_{T_k} \mathcal{L}_{TS}(T) := -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{J} 1\{y_i = j\} \cdot \log(\text{Softmax}(f(x_i)/T_k)_j),
$$

which optimizes the temperature parameter $T$ such that the negative log likelihood is minimized. We use $\text{TS}-\text{Alg}$ to denote the temperature scaling learning algorithm; given inputs dataset $D$ and base model $f$, $\text{TS}-\text{Alg}$ outputs the learned temperature parameter by solving Eq. (2), e.g., $T = \text{TS}-\text{Alg}(D, f)$.

### 3 Multi-domain temperature scaling

We propose our algorithm—multi-domain temperature scaling—that aims to improve the calibration of the predictions of a model on each domain. One key observation is that if we apply temperature scaling to each domain separately, then TS is able to produce calibrated confidence on every domain. Therefore, the question becomes how to “aggregate” these temperature scaling models and learn one calibration model, denoted by $\hat{h}$, that has similar performance to the $k$-th calibration model $h_k$ evaluated on domain $k$ for every $k \in [K]$.

At a high level, we propose to learn a calibration model that maps samples from the input space $X$ to the temperature space $\mathbb{R}_+$. To start with, we learn the temperature parameter $\hat{T}_k$ for the base model $f$ on every domain $k$ by applying temperature scaling on $D_k$. Next, we apply the base deep model to compute feature embeddings of samples from different domains, and label feature embeddings from...
the $k$-th domain with $\hat{T}_k$. In particular, we construct $K$ new datasets, $\hat{D}_1, \ldots, \hat{D}_K$, where each dataset contains feature embeddings and temperature labels from one domain, i.e., $\hat{D}_k = \{(\Psi(x_{i,k}, T_k))_{i=1}^{n_k}\}$. Finally, we apply linear regression on these labeled datasets. In detail, our algorithm is as follows:

1. **Learn temperature scaling model for each domain.** For every domain $k$, we learn temperature $\hat{T}_k$ by applying temperature scaling on validation data $D_k = \{(x_{i,k}, y_{i,k})\}_{i=1}^{n_k}$ from $k$-th domain, i.e., $\hat{T}_k = \text{TS-Alg}(D_k, f)$ and TS-Alg denotes the TS algorithm.

2. **Learn linear regression of temperatures.** Extract the feature embeddings of the base deep model $f$ on each domain. Use $\Psi(x_{i,k}) \in \mathbb{R}^p$ to denote the feature embedding of the $i$-th sample from $k$-th domain. Then we learn $\hat{\theta}$ by solving the following optimization problem,

   $$\hat{\theta} = \arg\min_{\theta} \sum_{k=1}^{K} \sum_{i=1}^{n_k} (\langle \Psi(x_{i,k}), \theta \rangle - \hat{T}_k)^2.$$ 

3. **Predict temperature on unseen test samples.** Given an unseen test sample $\bar{x}$, we first compute the predicted temperature $\hat{T}$ using the learned linear model $\hat{T} = \langle \Psi(\bar{x}), \hat{\theta} \rangle$. Then we output the confidence estimate for sample $\bar{x}$ as

   $$\hat{\pi} = \max_{j} \left[ \text{Softmax}(f(\bar{x})/\hat{T}) \right].$$

We denote our proposed method by MD-TS (Multi-Domain Temperature Scaling). A presentation of the algorithm in pseudocode can be found in Algorithm 1, Appendix A.

We pause to consider the basic concept in more detail. The goal of our proposed algorithm is to predict the best temperature for samples from different several domains. In an ideal setting where the learned linear model $\hat{\theta}$ results in good calibration on every InD domain, we can expect that $\hat{\theta}$ will continue to yield good calibration on the OOD domain $\mathcal{P}$ when $\mathcal{P}$ is close to one or several InD domains. For example, $\mathcal{P}$ will work well if $\mathcal{P}$ is a mixture of the $K$ domains, i.e., $\mathcal{P} = \sum_{k=1}^{K} \alpha_k \mathcal{P}_k$ and $\alpha \in \Delta^{K-1}$. Regarding the algorithmic design, linear regression is one of the simplest models for solving the regression problem. It is computationally fast to learn such linear models as well as make predictions on new samples, making it attractive. We test alternative, more flexible, regression algorithms in Section 4 but do not observe significant gains over linear regression.

To illustrate how our proposed algorithm MD-TS performs differently from standard TS, we return to the ImageNet-C dataset. We compare the predicted temperature of our algorithm on new samples from domain $k$ with the temperature that results from running TS on domain $k$ alone. The results are summarized in Figure 2, where each circle corresponds to the mean predicted temperature on one InD domain. For each domain, we also visualize the standard deviation of the predicted temperatures for samples from that domain (the horizontal bar around each point). We find that our algorithm predicts the temperature quite well. Note that it does not have access to the domain index information of the fresh samples. By contrast, TS always uses the same temperature, regardless of the input point.

![Figure 2: Compare the predicted temperature to the learned temperature $\hat{T}_k$ on the $k$-th domain.](image)

4 Experiments

In this section, we present experimental results evaluating our proposed method, demonstrating its effectiveness on both in-distribution and out-of-distribution calibration. We focus on three real-world datasets, including ImageNet-C [Hendrycks and Dietterich, 2019]—a widely used robustness
We summarize the ECE results of different methods on three datasets in Table 1 and Figure 3. We use the Expected Calibration Error (ECE) as the main evaluation metric. Evaluation metrics. We evaluate ECE on both InD domains and OOD domains. Specifically, we evaluate the ECE of the samples from OOD domains for ECE evaluation. We consider multiple network architectures for evaluation, including ResNet-50 [He et al., 2016], ResNext-50 [Xie et al., 2017], DenseNet-121 [Huang et al., 2017], BiT-M-R50 [Kolesnikov et al., 2020], Efficientnet-b1 [Tan and Le, 2019], ViT-Small, and ViT-Base [Dosovitskiy et al., 2020]. Evaluation metrics. We evaluate different calibration methods on three datasets, ImageNet-C, WILDS-RxRx1, and GLDv2. ImageNet-C contains 15 types of common corruptions where each corruption includes five severity levels. Each corruption with one severity is one domain, and there are 76 domains in total (including the standard ImageNet validation dataset). We partition the 76 domains into disjoint in-distribution domains and out-of-distribution by severity level or corruption type. WILDS-RxRx1 is a domain generalization dataset, and we treat each experimental domain as one domain. We adopt the default val/test split in Koh et al. [2021]: use the four validation domains as in-distribution domains and the remaining 32 domains as out-of-distribution domains. WILDS-RxRx1 contains 76 domains into disjoint five severity levels. Each corruption with one severity is one domain, and there are 823 domains in total. We randomly select 500 domains for training the model, and then use the remaining 323 domains for evaluation denoted by validation domains. We further screen the validation domains by removing the domains with less than 300 data points. There are 44 domains after screening, and we use 30 domains as in-distribution domains and the remaining 14 domains as out-of-distribution domains. For all datasets, we randomly sample half of the data from in-distribution domains for calibration models and use the remaining samples for InD ECE evaluation. We use all the samples from OOD domains for ECE evaluation.

4.1 Main results

We summarize the ECE results of different methods on three datasets in Table 1 and Figure 3. We use TS to denote temperature scaling [Guo et al., 2017], and use MSP to denote applying the maximum softmax probability [Hendrycks and Gimpel, 2016] of the model output (i.e., without calibration).
In Table 1, we use the ImageNet validation dataset and ImageNet-C datasets with severity level $s \in \{1, 5\}$ as the InD domains and use the remaining datasets as OOD domains. We present the averaged per-domain ECE results in Table 1, and visualize the ECE of each domain in Figure 3. As shown in Table 1 and Figure 3(a)-3(c), we find that our proposed approach achieves much better InD per-domain calibration compared with baselines. Also, TS does not significantly improve over MSP on ImageNet-C InD domains in Table 1, but our proposed method largely improve the ECE compared with MSP and TS. For instance, the ECE results of MSP and TS on Efficientnet-b1 are 6.93 and 6.99, and our method achieves 3.84. Intuitively, when there are a diverse set of domains in the calibration dataset, a single temperature cannot provide well-calibrated confidences. In contrast, our proposed method is able to produce much better InD confidence estimates by leveraging the multi-domain structure of the data.

Next we study the performance of different methods on out-of-distribution domains. From Table 1, we find that MD-TS achieves the best performance on OOD domains across all the settings. On ImageNet-C with BiT-M-R50, MD-TS improves the ECE from 6.54 (MSP) to 4.05, while the performance of TS is similar to MSP. Moreover, MD-TS significantly outperforms MSP and TS on WILDS-RxRx1, where MD-TS improves over TS by around 5.00 measured in ECE. Figure 3(d)-3(f) display the per-domain ECE performance on out-of-distribution domains. MD-TS improves over TS on more than half of the domains in all three datasets. For the remaining domains, MD-TS performs slightly worse than TS. Furthermore, on those domains that TS performs poorly (ECE > 8), MD-TS largely improves over TS by large margins. Further comparisons in Appendix C.6 show that these improvements continue to hold when relative to two other calibration techniques: MC dropout [Gal and Ghahramani, 2016] and deep ensembles [Lakshminarayanan et al., 2017].

### 4.2 Predicting generalization

Suppose a model can produce calibrated confidences on unseen samples, in which case we could leverage the calibrated confidence to predict the model performance. Specifically, based on the definition of ECE in Eq. (1), when the model is well-calibrated, the average of the calibrated confidence is close to the average accuracy, i.e., $\text{Conf}(\mathcal{D}) \approx \text{Acc}(\mathcal{D})$.\(^2\) Meanwhile, predicting model

\[^2\text{Conf}(\mathcal{D})\] denotes the average (calibrated) confidence on dataset $\mathcal{D}$, and $\text{Acc}(\mathcal{D})$ denotes the average accuracy on dataset $\mathcal{D}$.\]
Table 2: Model performance prediction comparison results of different methods on three datasets. Lower MAE indicates better performance.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Architectures</th>
<th>InD-domains MAE</th>
<th>OOD-domains MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSP</td>
<td>TS</td>
<td>MD-TS</td>
</tr>
<tr>
<td>ImageNet-C</td>
<td>ResNet-50</td>
<td>5.88</td>
<td>4.74</td>
</tr>
<tr>
<td></td>
<td>Bit-M-R50</td>
<td>6.08</td>
<td>6.16</td>
</tr>
<tr>
<td>WILDS-RxRx1</td>
<td>ResNet-50</td>
<td>33.65</td>
<td>9.61</td>
</tr>
<tr>
<td></td>
<td>ResNext-50</td>
<td>25.32</td>
<td>8.55</td>
</tr>
<tr>
<td></td>
<td>Bit-M-R50</td>
<td>12.67</td>
<td>7.18</td>
</tr>
</tbody>
</table>

Figure 4: Predicting accuracy performance of MD-TS and TS on both out-of-distribution domains. Each plot is shown with predicted accuracy (X-axis) and accuracy (Y-axis). Each points corresponds to one domain. The network architecture is ResNet-50 for three datasets. Point closer to the \( Y = X \) dashed line means better prediction performance.

4.3 MD-TS ablations

To learn a calibration model that performs well per-domain, we apply linear regression on feature representations \( \Phi(x_k) \) such that \( \langle \Phi(x_k), \theta \rangle \approx \hat{T}_k \), where \( x_k \) is from domain \( k \) and \( \hat{T}_k \) is the temperature parameter for domain \( k \). We investigate other methods for learning the map from feature representations to temperatures in a regression framework. Specifically, beside the ordinary least squares (OLS) used in Algorithm 1, we consider ridge regression (Ridge), robust regression with Huber loss (Huber), kernel ridge regression (KRR), and \( K \)-nearest neighbors regression (KNN). The implementations are mainly based on scikit-learn [Pedregosa et al., 2011]. We use grid search (on InD domains) to select hyperparameters for Ridge, Huber, KRR, and KNN.

We summarize the comparative results for different regression algorithms in Table 3. Compared to OLS, other regression algorithms do not achieve significant improvement. Specifically, KRR achieves slightly better performance on OOD domains, while other algorithms have similar performance compared to OLS. Moreover, there are no hyperparameter in OLS, which makes it more practical in
we first introduce the H
The Table 3: Per-domain ECE (\( \{ \)}
-domains) can lead to good OOD performance. To quantify the distance between two distributions,
generalization, where they show that minimizing the differences in training risks (w.r.t. different
et al., 2021] consider a similar multiple training domains setting in the context of out-of-distribution
\( h \) per-domain ECE on InD and OOD domains, and report the mean and standard error of per-domain
ECE. Lower ECE means better performance.

<table>
<thead>
<tr>
<th>Architectures</th>
<th>InD-domains</th>
<th>OOD-domains</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OLS</td>
<td>Ridge</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>2.85</td>
<td>2.88</td>
</tr>
<tr>
<td>ResNext-50</td>
<td>3.13</td>
<td>3.14</td>
</tr>
<tr>
<td>DenseNet-121</td>
<td>2.94</td>
<td>3.03</td>
</tr>
</tbody>
</table>

real-world problems. Meanwhile, the results suggest that our proposed MD-TS is stable to the choice
of specific regression algorithms.

5 Theoretical analysis

In this section, we provide theoretical analysis to support our understanding of our proposed algorithm
in the presence of distribution shifts. We use \( h^*_\alpha(\cdot) = h(\cdot; f, \beta^*_\alpha) : \mathcal{X} \to [0, 1] \) to denote the best
calibration map for the base model \( f \) on the \( k \)-th domain; this map \textit{minimizes} the expected calibration
error (ECE) \( \mathbb{E}[\|p - \mathbb{P}(\hat{y} = y)\| \tilde{p} = p)] \) over distribution \( \mathcal{P}_k \). We also call \( h^*_\alpha \) a hypothesis in the
hypothesis class \( \mathcal{H} \). Next, given the fixed base model \( f \), we aim to learn \( \hat{h}(\cdot) = h(\cdot; f, \beta) \) such that \( \varepsilon(h, \mathcal{P}_{k,X}) = \mathbb{E}_{X \sim \mathcal{P}_{k,X}}[|h^*_k(X) - \hat{h}(X)|] \) is small for every domain \( k \), where \( \varepsilon_k(h) \) denotes the risk of \( \hat{h} \) w.r.t. the the best calibration map \( h^*_k \) under domain \( \mathcal{P}_k \). In addition, we are interested in
generalizing to new domains: suppose there is an unseen OOD domain \( \mathcal{P} \) and its marginal feature
distribution is different from existing domains, i.e., \( \mathcal{P}_X \neq \mathcal{P}_{k,X} \) for \( k \in [K] \).

Our goal is to understand the conditions under which \( \hat{h} \) can have similar calibration on OOD domains as
the InD domains. For example, if the OOD domain is similar to the mixture distribution of InD
domains, we would expect \( \hat{h} \) performs similarly on InD and OOD domains. Previous work [Krueger
et al., 2021] consider a similar multiple training domains setting in the context of out-of-distribution
generalization, where they show that minimizing the differences in training risks (w.r.t. different
domains) can lead to good OOD performance. To quantify the distance between two distributions,
we first introduce the \( \mathcal{H} \)-divergence [Ben-David et al., 2010] to measure the distance between two
distributions:

\textbf{Definition 5.1 (\( \mathcal{H} \)-divergence).} \textit{Given an input space \( \mathcal{X} \) and two probability distributions \( \mathcal{P}_X \) and
\( \mathcal{P}_X' \) on \( \mathcal{X} \), let \( \mathcal{H} \) be a hypothesis class on \( \mathcal{X} \), and denote by \( \mathcal{A} \) the collection of subsets of \( \mathcal{X} \) which are the support of hypothesis \( h \in \mathcal{H} \), i.e., \( \mathcal{A}_{\mathcal{H}} = \{h^{-1}(1) \mid h \in \mathcal{H}\} \). The distance between \( \mathcal{P}_X \) and \( \mathcal{P}_X' \) is defined as
\( d_{\mathcal{H}}(\mathcal{P}_X, \mathcal{P}_X') = \sup_{A \in \mathcal{A}_{\mathcal{H}}}|\mathbb{P}_{\mathcal{P}_X}(A) - \mathbb{P}_{\mathcal{P}_X'}(A)| \).}

The \( \mathcal{H} \)-divergence reduces to the standard total variation (TV) distance when \( \mathcal{H} \) contains all measurable functions on \( \mathcal{X} \), which implies that the \( \mathcal{H} \)-divergence is upper bounded by the TV-distance, i.e., \( d_{\mathcal{H}}(\mathcal{P}_X, \mathcal{P}_X') \leq d_{TV}(\mathcal{P}_X, \mathcal{P}_X') \). On the other hand, when the hypothesis class \( \mathcal{H} \) has a finite VC
dimension or pseudo-dimension, the \( \mathcal{H} \)-divergence can be estimated using finite samples from \( \mathcal{P}_X \)
and \( \mathcal{P}_X' \) [Ben-David et al., 2010]. Next, we define the mixture distribution of the \( K \) in-distribution
domains \( \mathcal{P}_{k,X} \) on input space \( \mathcal{X} \) as follows:
\( \mathcal{P}_{k,X} = \sum_{k=1}^{K} \alpha_k \mathcal{P}_{k,X}, \quad \text{where} \quad \sum_{k=1}^{K} \alpha_k = 1 \text{ and } \alpha_k \geq 0. \)

Given multiple domains \( \{\mathcal{P}_1, \ldots, \mathcal{P}_K\} \), we can optimize the combination parameters \( \alpha \) such that
\( \mathcal{P}_{k,X} \) minimizes the \( \mathcal{H} \)-divergence between \( \mathcal{P}_{k,X} \) and \( \mathcal{P}_X \). More specifically, we define \( \hat{\alpha} \) as
\[ \hat{\alpha} = \arg\min_{\alpha \in \Delta} \left\{ \frac{1}{2} d_{\mathcal{H}}(\mathcal{P}_{k,X}, \mathcal{P}_X) + \lambda(\mathcal{P}_{k,X}, \mathcal{P}_X) \right\}, \quad \lambda(\mathcal{P}_{k,X}, \mathcal{P}_X) = \varepsilon(h^*, \mathcal{P}_{k,X}) + \varepsilon(h^*, \mathcal{P}_X). \]
where \( h^* := \arg\min_{h \in \mathcal{H}} \{ \varepsilon(h, \mathcal{P}_k^X) + \varepsilon(h, \bar{\mathcal{P}}_X) \} \) and \( \mathcal{H} \) is defined as \( \mathcal{H} := \{ \text{sign}(|h(x) - h'(x)| - t) \mid h, h' \in \mathcal{H}, 0 \leq t \leq 1 \} \). We now give an upper bound on the risk on the unseen OOD domain. This result follows very closely those of Blitzer et al. [2007]; Zhao et al. [2018], instantiated in our calibration setup. Details can be found in Appendix D.

**Theorem 5.2.** Let \( \mathcal{H} \) be a hypothesis class that contains functions \( h : \mathcal{X} \rightarrow [0,1] \) with pseudo-dimension \( \text{Pdim}(\mathcal{H}) = d \). Let \( \{D_{k,X}\}_{k=1}^{K} \) denote the empirical distributions generated from \( \{\mathcal{P}_{k,X}\}_{k=1}^{K} \), where \( D_{k,X} \) contains \( n \) i.i.d. samples from the marginal feature distribution \( \mathcal{P}_{k,X} \) of domain \( k \). Then for \( \delta \in (0,1) \), with probability at least \( 1 - \delta \), we have

\[
\varepsilon(h, \bar{\mathcal{P}}_X) \leq \sum_{k=1}^{K} \hat{\alpha}_k \cdot \varepsilon(h, D_{k,X}) + \frac{1}{2} \text{\bar{d}}_{\mathcal{H}}(\mathcal{P}_{k,X}^\alpha, \bar{\mathcal{P}}_X) + \lambda(\mathcal{P}_{k,X}^{\alpha}, \bar{\mathcal{P}}_X) + \tilde{O}\left(\frac{\text{Pdim}(\mathcal{H})}{\sqrt{nK}}\right), \tag{4}
\]

where \( \hat{\alpha} \) and \( \lambda(\mathcal{P}_{k,X}^{\alpha}, \bar{\mathcal{P}}_X) \) are defined in Eq. (3), \( \bar{\mathcal{P}}_X \) denotes the marginal distribution of the OOD domain, \( \text{Pdim}(\mathcal{H}) \) is the pseudo-dimension of the hypothesis class \( \mathcal{H} \), and \( \varepsilon(h, D_{k,X}) \) is the empirical risk of the hypothesis \( h \) on \( D_{k,X} \).

**Remark 5.3.** As shown in Theorem 5.2, even if the OOD domain is very different from the in-distribution domains, the Eq. (4) still implies that we could decrease the risk upper bound on the OOD domain if we perform multi-domain calibration. More specifically, if we could achieve good calibration performance on each individual domain by using multi-domain calibration (which that the first term in the RHS of Eq. (4) is small), then the term \( \frac{1}{2} \text{\bar{d}}_{\mathcal{H}}(\mathcal{P}_{k,X}^\alpha, \bar{\mathcal{P}}_X) + \lambda(\mathcal{P}_{k,X}^{\alpha}, \bar{\mathcal{P}}_X) \) is always smaller or equal to \( \frac{1}{2} \text{\bar{d}}_{\mathcal{H}}(\mathcal{P}', \bar{\mathcal{P}}_X) + \lambda(\mathcal{P}', \bar{\mathcal{P}}_X) \), where \( \mathcal{P}' \) is the pooled distribution or any individual domain distribution.

**Remark 5.4.** As suggested by Eq. (4) of Theorem 5.2, larger risks on in-distribution domains will lead to a larger upper bound for the risk evaluated on the OOD domain. On the other hand, as shown in Figure 1, a universal temperature is not sufficient to achieve good calibration performance on each individual in-distribution domain. Therefore, even in the mixture of in-distribution domain setting, a universal temperature is suboptimal and applying multi-domain temperature scaling could be better than using a universal temperature.

This result means that if we can learn a hypothesis \( \hat{h} \) that achieves small empirical risk \( \varepsilon(\hat{h}, D_{k,X}) \) on every domain, then \( \hat{h} \) is able to achieve good performance on the OOD domain if distribution of the OOD domain is similar to the mixture distribution of InD domains measured by \( \mathcal{H} \)-divergence. In this case, if the learned calibration map \( \hat{h} \) is well-calibrated on every domain \( \mathcal{P}_k \), then \( \hat{h} \) is likely to provide calibrated confidence for the OOD domain \( \bar{\mathcal{P}} \). Recall from Section 4, we proposed an algorithm that performs well across InD domains. The upper bound in Eq. (4) provides insight into understanding why this algorithm is effective.

### 6 Discussion

We have developed an algorithm for robust calibration that exploits multi-domain structure in datasets. Experiments on real-world domains indicate that multi-domain calibration is an effective way to improve the robustness of calibration under distribution shifts. Our proposed algorithm still needs validation domains to achieve strong calibration performance on OOD domains, one interesting direction for future work would be to extend our algorithm to a scenario where no domain information is available. We hope the multi-domain calibration perspective in this paper can motivate further work to close the gap between in-distribution and out-of-distribution calibration.

**Acknowledgments and Disclosure of Funding**

We would like to thank the anonymous reviewers for their constructive suggestions and comments. Yaodong Yu acknowledges support from the joint Simons Foundation-NSF DMS grant #2031899. Stephen Bates was supported by the Foundations of Data Science Institute and the Simons Institute. Yi Ma acknowledges support from ONR grants N00014-20-1-2002 and N00014-22-1-2102 and the joint Simons Foundation-NSF DMS grant #2031899. Michael Jordan acknowledges support from the ONR Mathematical Data Science program.
References


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Theorem 5.2.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix D.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4 and Appendix A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 4.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]
Appendix

A Additional Details

Pseudocode for MD-TS. We first provide additional details on our proposed algorithm, MD-TS (Multi-Domain Temperature Scaling), in Algorithm 1.

Algorithm 1 MD-TS

**Input:** Data from $k$-th domain $D_k = \{(x_{i,k}, y_{i,k})\}_{i=1}^{n_k}$, $k \in [K]$, base model $f$, feature embedding map of base model $\Psi$, and the test sample $\bar{x}$.

1: for $k = 1, \ldots, K$ do
2: $\hat{T}_k = \text{TS-Alg}(D_k, f)$
3: end for
4: Learn the linear model, $\hat{\theta} = \arg\min_{\theta} \sum_{k=1}^{K} \sum_{i=1}^{n_k} \left(\langle \Psi(x_{i,k}), \theta \rangle - \hat{T}_k \right)^2$.
5: Predict temperature $\bar{T}$ the test sample $\bar{x}$ using the learned linear model, $\bar{T} = \langle \Psi(x_{i,k}), \hat{\theta} \rangle$.
6: Compute the confidence estimate for sample $\bar{x}$ as $\bar{\pi} = \max_j \left[\text{Softmax}(f(\bar{x})/\bar{T})\right]_j$.

**Output:** Confidence estimate $\bar{\pi}$ for the test sample $\bar{x}$.

Experimental details (checklist). We provide additional details about the training and compute. Details about data splits and hyperparameters for training can be found in Section 4 and Appendix C. We use NVIDIA 2080Ti and A100 GPUs, and our experiments required around 100 hours of GPU time.

B Societal Impact

In this paper, we aim to improve the trustworthiness of machine learning systems by first, explicitly accounting for uncertainty, and second, do this in a way that is robust to distribution shifts. As uncertainty quantification is an increasingly important component of real-world machine learning systems, including health care and autonomous driving, we believe our work could potentially benefit a wide range of societal activities. Moreover, our method explicitly takes into account subgroups of the data, trying to achieve good performance across all data. It is known that this is an important aspect of performance when deploying models in high-consequence settings [Barocas et al., 2019]. We hope our work could offer a new perspective on uncertainty quantification under distributional shifts. We do not anticipate the negative social impact of this work.
C Additional Experimental Results

In this section, we provide additional implementation details and experimental results.

Details about dataset and pre-trained model. For ImageNet, we consider a 200 classes subset of ImageNet. Details can be found in Hendrycks et al. [2021]. The Efficientnet-b1 is trained by AdvProp and AutoAugment data augmentation [Xie et al., 2020].

C.1 Additional experimental results of predicting model accuracy

We provide additional results (other network architectures) of the prediction performance of TS and MD-TS on every OOD domain in Figure 5.

Figure 5: (Evaluated on more network architectures) Predicting accuracy performance of MD-TS and TS on both out-of-distribution domains. Each plot is shown with predicted accuracy (X-axis) and accuracy (Y-axis). Each point corresponds to one domain. The network architecture is ResNet-50 for three datasets. Point closer to the $Y = X$ dashed line means better prediction performance.
C.2 Additional experimental results of WILDS-RxRx1

We provide additional MDECE results of TS and MD-TS on WILDS-RxRx1 (evaluated on other InD/OOD splits) in Figure 6. For every InD/OOD split, we randomly sample 4 domains as the InD domains and set the remaining domains as OOD domains.

Figure 6: (Evaluated on WILDS-RxRx1 with other InD/OOD splits.) Per-domain ECE of MD-TS and TS on both in-distribution domains and out-of-distribution domains. Each plot is shown with ECE of TS (X-axis) and ECE of MD-TS (Y-axis). Top: per-domain ECE evaluated on InD domains. Bottom: per-domain ECE evaluated on OOD domains. Lower ECE is better.

C.3 Experimental results of overall ECE

We provide the overall ECE results, evaluated on the pooled InD/OOD data, of different methods on three datasets in Table 4.

Table 4: ECE (%) comparison on three datasets. We evaluate the ECE on pooled InD and OOD domains. We report the mean and standard error of ECE on one dataset. Lower ECE means better performance.
C.4 Experimental results of other calibration methods

We provide the MDECE results of other calibration methods, including histogram binning (Hist-Bin) [Zadrozny and Elkan, 2001], isotonic regression (Isotonic) [Zadrozny and Elkan, 2002], and Bayesian Binning into Quantiles (BBQ) [Naeini et al., 2015], on three datasets in Table 5. By comparing the results in Table 5 and Table 1, we find that our algorithm largely outperforms these methods on three datasets.

Table 5: Per-domain ECE (%) comparison of histogram binning (HistBin), isotonic regression (Isotonic), and Bayesian Binning into Quantiles (BBQ) on three datasets. We evaluate the per-domain ECE on InD and OOD domains. We report the mean and standard error of per-domain ECE on one dataset. Lower ECE means better performance.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Architectures</th>
<th>InD-domains</th>
<th>OOD-domains</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HistBin</td>
<td>Isotonic</td>
<td>BBQ</td>
</tr>
<tr>
<td>ImageNet-C</td>
<td>ResNet-50</td>
<td>9.50±0.25</td>
<td>5.17±0.12</td>
</tr>
<tr>
<td></td>
<td>Efficientnet-b1</td>
<td>7.10±0.22</td>
<td>5.30±0.06</td>
</tr>
<tr>
<td></td>
<td>BiT-M-R50</td>
<td>7.31±0.27</td>
<td>6.44±0.16</td>
</tr>
<tr>
<td></td>
<td>ViT-Base</td>
<td>6.85±0.26</td>
<td>4.30±0.03</td>
</tr>
<tr>
<td>WILDS-RxRx1</td>
<td>ResNet-50</td>
<td>11.64±0.23</td>
<td>11.67±0.42</td>
</tr>
<tr>
<td></td>
<td>ResNext-50</td>
<td>10.10±0.17</td>
<td>11.39±0.30</td>
</tr>
<tr>
<td></td>
<td>DenseNet-121</td>
<td>11.95±0.24</td>
<td>11.97±0.13</td>
</tr>
<tr>
<td>GLDv2</td>
<td>ResNet-50</td>
<td>14.77±0.16</td>
<td>11.43±0.07</td>
</tr>
<tr>
<td></td>
<td>Efficientnet-b1</td>
<td>6.71±0.05</td>
<td>4.41±0.09</td>
</tr>
<tr>
<td></td>
<td>BiT-M-R50</td>
<td>5.96±0.12</td>
<td>5.21±0.16</td>
</tr>
<tr>
<td></td>
<td>ViT-Small</td>
<td>3.72±0.05</td>
<td>3.78±0.06</td>
</tr>
</tbody>
</table>

C.5 Additional experimental results of ImageNet-C

We consider a different InD/OOD partition from the ones in Section 4. Specifically, we use the ImageNet validation dataset and ImageNet-C datasets with severity level \( s \in \{1, 2, 3, 4\} \) as the InD domains and use the remaining datasets as OOD domains. The results are summarized in Table 6. This is a more challenging setting since it requires calibration methods to extrapolate to a higher severity level. As shown in Table 6, our method still achieves the best performance in all settings, except for Efficientnet-b1 (OOD-domains). This is possibly because the Efficientnet-b1 model is pre-trained with AdvProp and AutoAugment data, it achieves better performance on corruptions with high severity levels than standard pre-trained models.

Table 6: Per-domain ECE (%) comparison on ImageNet-C datasets (with InD/OOD split mentioned in Section C.5). We evaluate the per-domain ECE on InD and OOD domains. We report the mean and standard error of per-domain ECE on one dataset. Lower ECE means better performance.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Architectures</th>
<th>InD-domains</th>
<th>OOD-domains</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSP</td>
<td>TS</td>
<td>MD-TS</td>
</tr>
<tr>
<td>ImageNet-C</td>
<td>ResNet-50</td>
<td>5.99±0.14</td>
<td>5.11±0.07</td>
</tr>
<tr>
<td></td>
<td>Efficientnet-b1</td>
<td>6.71±0.05</td>
<td>4.41±0.09</td>
</tr>
<tr>
<td></td>
<td>BiT-M-R50</td>
<td>5.96±0.12</td>
<td>5.21±0.16</td>
</tr>
<tr>
<td></td>
<td>ViT-Base</td>
<td>3.72±0.05</td>
<td>3.78±0.06</td>
</tr>
</tbody>
</table>
C.6 Compare to additional existing methods

We compare our proposed method with Ensembles [Lakshminarayanan et al., 2017] and Dropout [Gal and Ghahramani, 2016], and the results are summarized in Table 7.

Table 7: Per-domain ECE (%) comparison on two datasets. We evaluate the per-domain ECE on InD and OOD domains. We report the mean and standard error of per-domain ECE on one dataset. Lower ECE means better performance. We use Ensembles to denote the deep ensembles method proposed in Lakshminarayanan et al. [2017], and Dropout to denote the Monte-Carlo Dropout method proposed in Gal and Ghahramani [2016].

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Architectures</th>
<th>InD-domains</th>
<th>OOD-domains</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ensembles</td>
<td>Dropout</td>
<td>MD-TS</td>
</tr>
<tr>
<td>WILDS-RxRx1</td>
<td>ResNet-50</td>
<td>14.73±0.67</td>
<td>18.59±0.58</td>
</tr>
<tr>
<td>GLDv2</td>
<td>ResNet-50</td>
<td>10.85±0.08</td>
<td>11.73±0.13</td>
</tr>
</tbody>
</table>

We also study the performance of applying a 2-layer MLP, which has been used in Kumar and Sarawagi [2019], instead of the linear model in our proposed approach for predicting the temperature. We do not find that using 2-layer MLP lead to better performance compared to the linear model approach. For example, on WILDS-RxRx1 with ResNet-50, the per-domain ECE of the 2-layer MLP is 5.62% whereas per-domain ECE of the linear model is 5.25%.

C.7 Additional experimental results on the NLP dataset

We study the performance of our proposed method on WILDS-Amazon, and we present the results in Table 8 and Figure 7.

Table 8: Per-domain ECE (%) comparison on WILDS-Amazon [Koh et al., 2021]. We evaluate the per-domain ECE on InD and OOD domains. We report the mean and standard error of per-domain ECE on one dataset. Lower ECE means better performance.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Architectures</th>
<th>InD-domains</th>
<th>OOD-domains</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSP</td>
<td>TS</td>
<td>MD-TS</td>
</tr>
<tr>
<td>WILDS-Amazon</td>
<td>DistilBERT</td>
<td>31.88±0.22</td>
<td>10.16±0.22</td>
</tr>
</tbody>
</table>

Figure 7: (Evaluated on WILDS-Amazon.) Per-domain ECE of MD-TS and TS on both in-distribution domains and out-of-distribution domains. Each plot is shown with ECE of TS (X-axis) and ECE of MD-TS (Y-axis). Top: per-domain ECE evaluated on InD domains. Bottom: per-domain ECE evaluated on OOD domains. Lower ECE is better.
C.8 Additional results on efficiency of MD-TS

**Efficiency.** We study the efficiency of our proposed method by measuring running time on three datasets (seconds). We consider the ResNet50 for all datasets. By using the standard sklearn.linear_model, it takes 39.8s/3.2s/4.1s on ImageNet-C/WILDS-RxRx1/GLDv2 for solving the linear regression problem of MD-TS, and the overall running time of MD-TS is 49.7s/4.6s/6.6s on ImageNet-C/WILDS-RxRx1/GLDv2. standard TS takes 7.8s/1.2s/3.6s on ImageNet-C/WILDS-RxRx1/GLDv2. We summarize the comparison results in Table 9, Appendix C.8.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Architectures</th>
<th>RunTime (OLS of MD-TS)</th>
<th>Overall RunTime (MD-TS)</th>
<th>Overall RunTime (TS [Guo et al., 2017])</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet-C</td>
<td>ResNet-50</td>
<td>39.8s</td>
<td>49.7s</td>
<td>7.8s</td>
</tr>
<tr>
<td>WILDS-RxRx1</td>
<td>ResNet-50</td>
<td>3.2s</td>
<td>4.6s</td>
<td>1.2s</td>
</tr>
<tr>
<td>GLDv2</td>
<td>ResNet-50</td>
<td>4.1s</td>
<td>6.6s</td>
<td>3.6s</td>
</tr>
</tbody>
</table>

C.9 Additional experimental results on Brier Score

Table 10: Per-domain Brier Score [Brier et al., 1950] comparison on three datasets. We evaluate the per-domain Brier Score on InD and OOD domains. We report the mean and standard error of per-domain Brier Score on one dataset. Lower Brier Score means better performance.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Architectures</th>
<th>InD-domains</th>
<th>OOD-domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet-C</td>
<td>ResNet-50</td>
<td>0.533±0.018</td>
<td>0.527±0.017</td>
</tr>
<tr>
<td></td>
<td>Efficientnet-b1</td>
<td>0.430±0.016</td>
<td>0.429±0.016</td>
</tr>
<tr>
<td></td>
<td>BiT-M-R50</td>
<td>0.475±0.017</td>
<td>0.475±0.017</td>
</tr>
<tr>
<td></td>
<td>ViT-Base</td>
<td>0.327±0.014</td>
<td>0.327±0.014</td>
</tr>
<tr>
<td>WILDS-RxRx1</td>
<td>ResNet-50</td>
<td>1.053±0.019</td>
<td>0.908±0.009</td>
</tr>
<tr>
<td></td>
<td>ResNext-50</td>
<td>0.985±0.017</td>
<td>0.901±0.010</td>
</tr>
<tr>
<td></td>
<td>DenseNet-121</td>
<td>1.041±0.020</td>
<td>0.904±0.010</td>
</tr>
<tr>
<td>GLDv2</td>
<td>ResNet-50</td>
<td>0.788±0.003</td>
<td>0.787±0.002</td>
</tr>
<tr>
<td></td>
<td>BiT-M-R50</td>
<td>0.680±0.004</td>
<td>0.663±0.003</td>
</tr>
<tr>
<td></td>
<td>ViT-Small</td>
<td>0.670±0.004</td>
<td>0.665±0.003</td>
</tr>
</tbody>
</table>
D Missing Proofs

In this section, we present the proof for Theorem 5.2.

**Theorem D.1 (Restatement of Theorem 5.2).** Let $\mathcal{H}$ be a hypothesis class that contains functions $h : \mathcal{X} \to [0, 1]$ with pseudo-dimension $\text{Pdim}(\mathcal{H}) = d$. Let $\{D_{k,X}\}_{k=1}^K$ denote the empirical distributions generated from $\{(\mathcal{P}_{k,X})\}_{k=1}^K$, where $D_{k,X}$ contains $n$ i.i.d. samples from the marginal feature distribution $\mathcal{P}_{k,X}$ of domain $k$. Then for $\delta \in (0, 1)$, with probability at least $1 - \delta$, we have

$$
\varepsilon(h, \mathcal{P}_{k,X}) \leq \sum_{k=1}^K \alpha_k \cdot \varepsilon(h, \mathcal{D}_{k,X}) + \frac{1}{2} d_{\mathcal{H}}(\mathcal{P}_{K,X}^\alpha, \mathcal{P}_{X}) + \lambda(\mathcal{P}_{K,X}^\alpha, \mathcal{P}_X) + O\left(\frac{\text{Pdim}(\mathcal{H})}{\sqrt{nK}}\right),
$$

where $\alpha$ and $\lambda(\mathcal{P}_{k,X}^\alpha, \mathcal{P}_X)$ are defined in Eq. (3). $\mathcal{P}_X$ denotes the marginal distribution of the OOD domain. $\text{Pdim}(\mathcal{H})$ is the pseudo-dimension of the hypothesis class $\mathcal{H}$, and $\varepsilon(h, \mathcal{D}_{k,X})$ is the empirical risk of the hypothesis $h$ on $D_{k,X}$.

Above, we use $\bar{O}(\cdot)$ to mean $O(\cdot)$ with some additional poly-logarithmic factors.

**Proof.** To start with, we use $\varepsilon(h, h', \mathcal{P}_X)$ to denote $\varepsilon(h, h', \mathcal{P}_X) = \mathbb{E}_{X \sim \mathcal{P}_X}[|h(X) - h'(X)|]$. Let $\varepsilon(h, \mathcal{P}_X)$ denote $\varepsilon(h, h^*, \mathcal{P}_X)$, where $h^*$ minimizes the risk under $\mathcal{P}_X$. We can upper bound $\varepsilon(h, h^*, \mathcal{P}_X)$ by

$$
\varepsilon(h, h^*, \mathcal{P}_X) \leq \varepsilon(h^*, h^*, \mathcal{P}_X) + \varepsilon(h, h^*, \mathcal{P}_X) \leq \varepsilon(h^*, h^*, \mathcal{P}_X) + \varepsilon(h, h^*, \mathcal{P}_X) - \varepsilon(h, h^*, \mathcal{P}_X) + \varepsilon(h, h^*, \mathcal{P}_X) \leq \varepsilon(h^*, h^*, \mathcal{P}_X) - \varepsilon(h, h^*, \mathcal{P}_X) + \varepsilon(h, h^*, \mathcal{P}_X)
$$

by the triangle inequality (see Lemma 3 of Zhao et al. [2018]). Above, $h^*$ is defined as

$$
h^* := \text{argmin}_{h \in \mathcal{H}} \{\varepsilon(h, \mathcal{P}_{k,X}) + \varepsilon(h, \mathcal{P}_X)\}.
$$

Next, we have

$$
\varepsilon(h, h^*, \mathcal{P}_X) \leq \varepsilon(h^*, h^*, \mathcal{P}_X) + \varepsilon(h, h^*, \mathcal{P}_X) \leq \varepsilon(h^*, h^*, \mathcal{P}_X) + \frac{1}{2} d_{\mathcal{H}}(\mathcal{P}_{k,X}^\alpha, \mathcal{P}_X) + \frac{1}{2} d_{\mathcal{H}}(\mathcal{P}_{K,X}^\alpha, \mathcal{P}_X) + \varepsilon(h, h^*, \mathcal{P}_X) \leq \varepsilon(h^*, h^*, \mathcal{P}_X) + \frac{1}{2} d_{\mathcal{H}}(\mathcal{P}_{k,X}^\alpha, \mathcal{P}_X) + \varepsilon(h, h^*, \mathcal{P}_X) \leq \varepsilon(h^*, h^*, \mathcal{P}_X) + \frac{1}{2} d_{\mathcal{H}}(\mathcal{P}_{k,X}^\alpha, \mathcal{P}_X) + \varepsilon(h, h^*, \mathcal{P}_X)
$$

where the last step is using the Lemma 1 in Zhao et al. [2018]. Therefore, we have

$$
\varepsilon(h, h^*, \mathcal{P}_X) \leq \varepsilon(h^*, h^*, \mathcal{P}_X) + \frac{1}{2} d_{\mathcal{H}}(\mathcal{P}_{k,X}^\alpha, \mathcal{P}_X) + \varepsilon(h, h^*, \mathcal{P}_X) \leq \varepsilon(h^*, h^*, \mathcal{P}_X) + \frac{1}{2} d_{\mathcal{H}}(\mathcal{P}_{k,X}^\alpha, \mathcal{P}_X) + \varepsilon(h, h^*, \mathcal{P}_X) \leq \varepsilon(h^*, h^*, \mathcal{P}_X) + \frac{1}{2} d_{\mathcal{H}}(\mathcal{P}_{k,X}^\alpha, \mathcal{P}_X) + \varepsilon(h, h^*, \mathcal{P}_X)
$$

where we apply the triangle inequality in second step and $h^*_{K,\alpha}$ minimizes the risk under $\mathcal{P}_{k,X}^\alpha$. Next, by Theorem 10.6 in Mohri et al. [2018], Theorem 1.2 and Lemma 2 in Zhao et al. [2018], we have for $\delta \in (0, 1)$, with probability at least $1 - \delta$,

$$
\varepsilon(h, h^*_{K,\alpha}, \mathcal{P}_{k,X}) \leq \varepsilon(h, h^*_{K,\alpha}, \mathcal{P}_{k,X}) + \bar{O}\left(\frac{\text{Pdim}(\mathcal{H})}{\sqrt{nK}}\right).
$$

Meanwhile, based on its definition, $\bar{O}(h, h^*_{K,\alpha}, \mathcal{P}_{k,X})$ can be rewritten as

$$
\bar{O}(h, h^*_{K,\alpha}, \mathcal{P}_{k,X}) = \sum_{k=1}^K \alpha_k \cdot \bar{O}(h, h^*_{k}, \mathcal{P}_{k,X}).
$$

21
Putting Eq. (5), (6), and (7) together, we have for $\delta \in (0, 1)$, with probability at least $1 - \delta$,

$$\varepsilon(h, \bar{P}_X) \leq \sum_{k=1}^{K} \alpha_k \cdot \hat{\varepsilon}(h, D_{k,X}) + \frac{1}{2} d_H(\bar{P}_k, \bar{P}_X) + \lambda(\bar{P}_k, \bar{P}_X) + \tilde{O} \left( \frac{\text{Pdim}(\mathcal{H})}{\sqrt{nK}} \right)$$

hold for any $\alpha \in \Delta$ and $h \in \mathcal{H}$. Thus, we complete our proof.

$\Box$