# Appendix for: On the Pitfall of Mixup Training for Uncertainty Calibration

## Contents


**1. Calibration Metrics** 1

**2. Temperature Scaling** 2

**3. Details of our Approaches** 2
- 3.1. Mixup Inference . . . 2
- 3.2. Mixup Inference in Training . . . 3
- 3.3. Implementations . . . 4

**4. Additional Experiment Results** 4
- 4.1. Results on other Metrics . . . 4
- 4.2. Comparison with other Calibration Methods . . . 7
- 4.3. Impact of the Hyperparameter . . . 9
- 4.4. Results with other Settings . . . 9


## 1. Calibration Metrics

Except for ECE, ACE and NLL mentioned in the paper, we have further evaluated models on two other calibration metrics: Adaptive Expected Calibration Error (AdaECE) [12] and Thresholded Expected Calibration Error (TECE) [12]. We formally introduce them as follows.

**ECE.** As we discussed in Subsection 2.2 in paper, a perfectly calibrated model should satisfy $\mathbb{P}(\hat{y}=y \mid \hat{p}=p)=p$ for $p \in [0,1]$. Given this, calibration performance could be measured by the difference between accuracy and confidence in expectation, i.e., $\mathbb{E}_{\hat{p}}\left[\mathbb{P}(\hat{y}=y \mid \hat{p}=p)-p\right]$. In practice, this can be approximated by first grouping all the samples into $M$ equally spaced bins $\{B_m\}_{m=1}^{M}$ with respect to their confidence scores, and taking a weighted average of the accuracy/confidence difference between these bins [1]. Formally, ECE is defined as [9]:

$$\mathrm{ECE}=\sum_{m=1}^{M}\frac{|B_m|}{N}\left|\mathrm{acc}\left(B_m\right)-\mathrm{avg\,Conf}\left(B_m\right)\right|,$$

where $N$ denotes the total number of samples in testing set.

**ACE.** By replace the weighting term in ECE with a uniform weighting factor, ACE can be formally defined as [11]:

$$\mathrm{ACE}=\frac{1}{M}\sum_{m=1}^{M}\left|\mathrm{acc}\left(B_m\right)-\mathrm{avg\,Conf}\left(B_m\right)\right|.$$

**AdaECE.** Different from ECE that bins with equal confidence interval, AdaECE adaptively groups the samples into intervals with same sample size. In this way, each bin $B_r$ in $\{B_r\}_{r=1}^{R}$ has $N/M$ samples and the metric can be formally defined as [12]:

$$\mathrm{AdaECE}=\frac{1}{R}\sum_{r=1}^{R}\left|\mathrm{acc}\left(B_r\right)-\mathrm{avg\,Conf}\left(B_r\right)\right|.$$

[1] In our experiments, we set the bin number as 15 for all calibration metrics.

**TECE.** Practitioner may only focus on the calibration of the predictions with confidence scores above a given threshold. To this end, TECE evaluates by firstly filtering out the samples with confidence below a specified threshold, and then calculating ECE on the remaining samples [12].

**NLL.** The negative log-likelihood is also commonly used to measure a model's prediction quality [3]. In multi-class classification setting, NLL is equivalent to cross-entropy, which is commonly used as a loss function in deep learning. Formally, NLL is defined as:

$$\text{NLL} = -\frac{1}{N}\sum_{i=1}^{N}\log(p(y_i|x_i)),$$

where $p(y_i|x_i)$ denotes the model's output confidence on the ground-truth label $y_i$. It should be noted that different from other calibration metrics, NLL takes the ground-truth label of each individual sample into account in the evaluation, and hence would be influenced by the model's predictive power. For example, even with zero ECE, the model cannot achieve good NLL with poor accuracy.

## 2. Temperature Scaling

Due to the simplicity and impressive generalization performance, Temperature Scaling (TS) is the most commonly adopted post-calibration method. By scaling the logits produced by a learned model with a temperature $\tau$, the sharpness of output probabilities can be significantly changed. Formally, after TS, model's confidence can be expressed as:

$$\hat{p} = \max_{i\in[K]} \frac{\exp\left(\delta_i/\tau\right)}{\sum_{k=1}^{K}\exp\left(\delta_i/\tau\right)},$$

where $\delta_i$ means the $i$-th logit element. The temperature $\tau$ softens the output probability with $\tau > 1$ and sharpens the probability with $\tau < 1$. As $\tau \to 0$, model's outputs collapse to a one-hot vector. As $\tau \to \infty$, model's outputs converge to a uniform vector. In practice, we need to first find the temperature that yields the best calibration result on a hold-out validation set, and then apply this temperature to the softmax layer.

## 3. Details of our Approaches

### 3.1. Mixup Inference

**Derivation of Equation (4).** As we presented in paper, under the basic assumption that linear interpolation of features leads to linear interpolation of the outputs, a mixup-trained model's outputs would satisfy the following equations:

$$\begin{aligned}\widehat{\widetilde{y}}_1 &= \lambda_1\widehat{y}_a + (1-\lambda_1)\widehat{y}_b,\\ \widehat{\widetilde{y}}_2 &= \lambda_2\widehat{y}_a + (1-\lambda_2)\widehat{y}_b,\end{aligned}$$

where $\widehat{y}_\alpha$ and $\widehat{\widetilde{y}}_i$ denote the model outputs of the original and mixed samples respectively. Then, simple derivations of this system of equations can be obtained:

$$\begin{aligned}\widehat{y}_a &= \frac{\widehat{\widetilde{y}}_1 - \widehat{\widetilde{y}}_2(1-\lambda_1)/(1-\lambda_2)}{\lambda_1 - \lambda_2(1-\lambda_1)/(1-\lambda_2)},\\ \widehat{y}_b &= \frac{\widehat{\widetilde{y}}_1 - \widehat{\widetilde{y}}_2\lambda_2/\lambda_1}{1-\lambda_2-(1-\lambda_1)\lambda_2/\lambda_1}.\end{aligned} \tag{1}$$

Therefore, by rewriting $\widehat{\widetilde{y}}_i$ as $f(\widetilde{x}_i)$, we obtain the decoupled outputs of $x_a$ and $x_b$ as shown by Equation (4) in the paper.

**Algorithm.** Algorithm 1 shows the pseudo-code of our mixup inference (MI) approach. The major difference between our MI approach and the MI approach used in [13] lies in the decoupling process as shown in line 7. To this end, we need to mix the inputs of samples twice before the forward pass, where we adopt the same coefficient $\alpha$ as is used in the training phase. Moreover, by setting $\lambda_2 = 0$, the outputs of $x_2$ can be collected before testing, and hence only one single forward pass is necessary in each iteration.

**Algorithm 1** Mixup Inference (MI)

**Input:** The model $f$; the sample $x$; the sample pool $\mathcal{S}$; the coefficient $\alpha$; the number of inference iterations $T$.
**Procedure:**
Initialize $\widehat{y} = 0$;

1: **for** $t = 1$ **to** $T$ **do**
2: Randomly select a sample $x'$ from $\mathcal{S}$;
3: Sample $\lambda_1$ and $\lambda_2$ form $\text{Beta}_{[0.5,1]}(\alpha, \alpha)$ and $\text{Beta}_{[0,0.5]}(\alpha, \alpha)$ respectively;
4: Mix sample $x$ with $x'$ twice as: $\widetilde{x}_1 = \lambda_1 x + (1-\lambda_1)x'$, $\widetilde{x}_2 = \lambda_2 x + (1-\lambda_2)x'$;
5: Forward pass of mixed samples: $\widehat{\widetilde{y}}_1 = f(\widetilde{x}_1)$, $\widehat{\widetilde{y}}_1 = f(\widetilde{x}_1)$;
6: Calculate $\widehat{y}_t$ by decoupling $\widetilde{y}_1, \widetilde{y}_1$ as $\widehat{y}_t = \frac{\widehat{\widetilde{y}}_1 - \widehat{\widetilde{y}}_2(1-\lambda_1)/(1-\lambda_2)}{\lambda_1 - \lambda_2(1-\lambda_1)/(1-\lambda_2)}$;
7: Update the logit output: $\widehat{y} = \widehat{y} + \frac{1}{T}\widehat{y}_t$;
8: **end for**

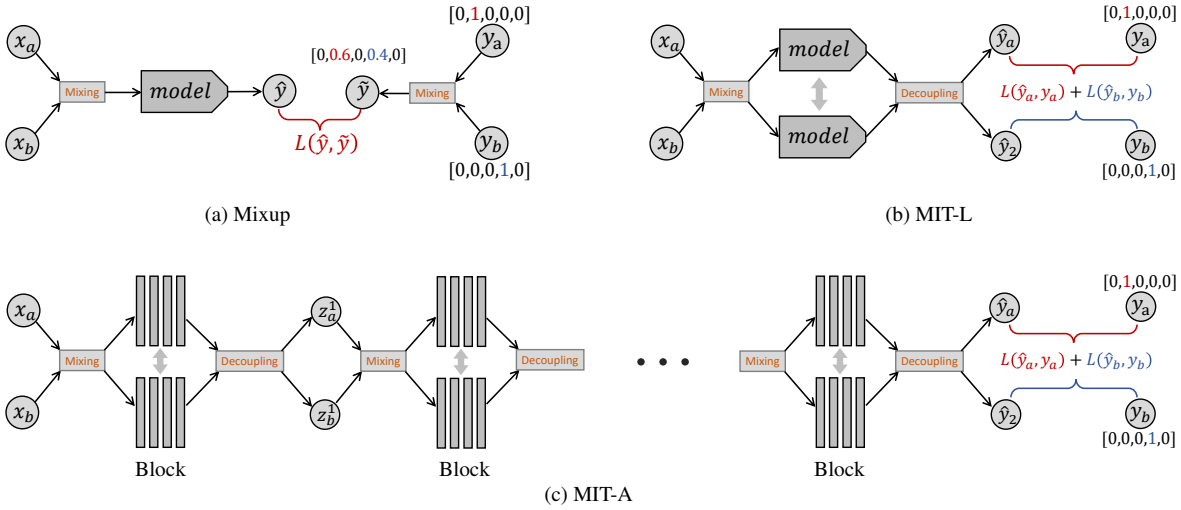
**Output:** The probabilistic output $\texttt{softmax}(\widehat{y})$.

### 3.2. Mixup Inference in Training

To employ our MIT strategy, one only needs to embed the mix-then-decouple procedure shown line 2-8 of Algorithm 1 into each mini-batch training process. We should note that there exists inevitable noise in models' outputs, and the noise would be enlarged in the decoupling process if the difference between $\lambda_1$ and $\lambda_2$ is small. Therefore, we propose to sample $\lambda_1$ and $\lambda_2$ from $\text{Beta}_{[0.5,1]}(\alpha, \alpha)$ and $\text{Beta}_{[0,0.5]}(\alpha, \alpha)$ respectively, or further force them to be greater than a specific constant. A Pytorch example of mini-batch training is shown as follows:

```
# Sampling λ1 and λ2 for mixup
l1 = np.random.beta(alpha, alpha)
l1 = max(l1, 1 - l1)
l2 = np.random.beta(alpha, alpha)
l2 = min(l2, 1 - l2)
while constraint == True and abs(l1 - l2) < margin:
    l1 = np.random.beta(alpha, alpha)
    l1 = max(l1, 1 - l1)
    l2 = np.random.beta(alpha, alpha)
    l2 = min(l2, 1 - l2)
# Mixing inputs by Equation (3)
mixed_inputs1 = l1 * inputs_a + (1 - l1) * inputs_b
mixed_inputs2 = l2 * inputs_a + (1 - l2) * inputs_b
# Forward Propagation
mixed_outputs1 = model(mixed_inputs1)
mixed_outputs2 = model(mixed_inputs2)
# Decoupling outputs by Equation (4)
outputs_a = (mixed_outputs1 - (1 - l1) / (1 - l2) * mixed_outputs2) / (l1 - l2 * (1 - l1) / (1 - l2))
outputs_b = (mixed_outputs2 - l2 / l1 * mixed_outputs1) / (1 - l2 - (1 - l1) * l2 / l1)
# Calculating Softmax Cross-Entropy loss
loss1 = criterion(outputs_a, targets_a)
loss2 = criterion(outputs_b, targets_b)
loss = 0.5 * loss1 + 0.5 * loss2
# Updating model's weights
optimizer.zero_grad()
loss.backward()
optimizer.step()
```

Moreover, the above mix-then-decouple process can be further embedded into the latent layers of neural networks. As we mentioned in the paper, MIT-A applies this process in each block and also the last layer of ResNets. As is shown in Figure A(c), the features produced by the first block are decoupled into $z_a^1, z_b^1$. Then, we remix $z_a^1, z_b^1$ with newly sampled $\lambda_1^1, \lambda_2^1$ and feed the remixed features into the next block. Most importantly, the logit outputs in the end of forward pass will be decoupled to approximately recover the $\widehat{y}_a$ and $\widehat{y}_b$ before calculating the softmax cross-entropy loss.




(a) Mixup (b) MIT-L

(c) MIT-A

Figure A. (a), (b) and (c) show the pipelines of vanilla mixup, our MIT-L and MIT-A respectively, where where loss function $L$ is the widely used softmax cross-entropy loss. For MIT-A, we simply apply the mix-then-decouple process in every block of ResNets. It should be noted that this process can also be applied in each latent layer.

### 3.3. Implementations

**Datasets.** Our experiments are conducted on 4 image classification datasets: SVHN [10], CIFAR10, CIFAR-100 [4] and Tiny-ImageNet [2]. In our experiments, we need hold-out validation sets to perform post-calibration, so we split each dataset into train/validation/test sets as following ratios: 68257/5k/26032 for SVHN, 45k/5k/10k for CIFAR-10/100 and 90k/10k/10k for Tiny-ImageNet.

**Optimization.** The implementation is based on PyTorch [14] and the experiments were carried out with NVIDIA Tesla V100 GPU. We use SGD as the opimizer with a momentum of 0.9, a weight decay of 1e-4. We train on SVHN and CIFAR-10/100 by total 350 epochs with the initial learning rate as 0.1, and divide it by a factor of 10 after 150 epochs and 250 epochs respectively. We train on Tiny-ImageNet by total 200 epochs with the initial learning rate as 0.01, and divide it by a factor of 10 after 100 epochs and 150 epochs respectively. We set the batch size as 128 on SVHN, CIFAR-10/100, and 64 on Tiny-ImageNet. The pretrained checkpoints used in our paper are provided by Pytorch official release [2]. Our code is available on GitHub: https://github.com/dengbaowang/Mixup-Inference-in-Training.

## 4. Additional Experiment Results

### 4.1. Results on other Metrics

Table A, D, B and C present the comparative results on calibrated ACE, AdaECE, TECE and NLL. Same with the main paper, the orange/blue color indicates that a method outperforms/underperforms ERM in average. The **boldface** and underline denote the best and the second best results of each row, and the marker † means the backbone is pretrained. The reported results are the average of 3 random runs, while in each run the results of last 10 epochs are averaged as the final result. As is shown in these tables, the results on these four calibration metrics are highly consistent to the results on ECE. On ACE, AdaECE and TECE ($\tau = 0.6$), our approaches, as well as two ablated mixup variants Mixup-SC and Mixup-IO, improve calibration performance compared with ERM. It is notable that if we do not consider the predictive performance, Mixup-SC and Mixup-IO even achieve larger improvements on these three metrics on average. On the contrary, vanilla mixup and two variants Mixup-DT, Mixup-TO underperform ERM in most cases. For NLL, since it also reflects the predictive performance, our approaches significantly outperform the others, including Mixup-SC and Mixup-IO. Figure B shows the comparative results in terms of the calibrated TECE with different choices of threshold $\tau$ on ResNet110. When $\tau$ becomes larger, more and more samples are discarded and only a few confident samples would be remained for evaluation. As is shown, our approaches achieve better results than most of the others. We can also observe that, without the constraint on $\Delta\lambda$ or the mixup process in latent layers (see **MIT-A** and **MIT-L** ($\Delta\lambda > \frac{1}{2}$)), our strategy still works well on calibration, while these techniques benefit the predictive accuracy. Table E shows the results with DenseNet121, where the overall results are

[2] https://pytorch.org/vision/stable/models.html

Table A. The overall comparative results in terms of the **Calibrated ACE**.

| | Backbones | ERM | Mixup (0.1) | Mixup (0.5) | Mixup (1.0) | Mixup (DT) | Mixup (TO) | Mixup (SC) | Mixup (IO) | **MIT-A** | **MIT-L** ($\Delta\lambda{>}\frac{1}{2}$) | **MIT-A** ($\Delta\lambda{>}\frac{1}{2}$) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SVHN | ResNet18 | 5.74 (8) | 5.39 (7) | 6.06 (9) | 6.36 (10) | 4.98 (5) | 7.27 (11) | 4.79 (2) | **4.76 (1)** | 5.00 (6) | 4.92 (3) | 4.98 (4) |
| | ResNet50 | 7.08 (5) | 10.5 (10) | 7.45 (8) | 7.26 (7) | 7.16 (6) | 11.3 (11) | **4.94 (1)** | 5.12 (2) | 6.35 (4) | 5.27 (3) | 7.90 (9) |
| | ResNet110 | 6.77 (5) | 6.82 (7) | 6.77 (6) | 7.26 (9) | 8.55 (10) | 14.1 (11) | 5.26 (2) | 6.26 (3) | 6.29 (4) | **5.01 (1)** | 7.25 (8) |
| | ResNet152 | 8.94 (7) | 12.9 (10) | 8.72 (6) | 10.3 (8) | 10.7 (9) | 13.0 (11) | 6.14 (3) | 4.93 (2) | 7.86 (5) | **4.31 (1)** | 7.16 (4) |
| | Avg. gain | — | + 1.80 | + 0.12 | + 0.67 | + 0.72 | + 4.33 | – 1.84 | – 1.86 | – 0.75 | – 2.25 | – 0.30 |
| CIFAR-10 | ResNet18 | **4.81 (1)** | 6.01 (3) | 7.73 (7) | 8.36 (9) | 7.09 (6) | 9.32 (11) | 5.11 (2) | 7.94 (8) | 6.02 (4) | 6.21 (5) | 8.45 (10) |
| | ResNet50 | 6.78 (4) | 9.59 (7) | 11.6 (10) | 10.5 (9) | 13.2 (11) | 9.87 (8) | 6.53 (2) | **5.97 (1)** | 6.63 (3) | 8.13 (6) | 7.36 (5) |
| | ResNet110 | 6.72 (3) | 11.2 (8) | 13.0 (10) | 10.6 (7) | 12.0 (9) | 15.6 (11) | 5.98 (2) | **5.39 (1)** | 8.18 (6) | 7.24 (4) | 7.70 (5) |
| | ResNet152 | 7.47 (5) | 9.69 (7) | 11.9 (9) | 13.8 (11) | 12.2 (10) | 10.8 (8) | **6.57 (1)** | 7.35 (4) | 6.72 (2) | 7.93 (6) | 7.19 (3) |
| | Avg. gain | — | + 2.69 | + 4.65 | + 4.39 | + 4.70 | + 4.97 | – 0.39 | + 0.21 | + 0.44 | + 0.93 | + 1.23 |
| CIFAR-100 | ResNet18 | 3.46 (9) | 2.69 (4) | **2.08 (1)** | 2.16 (2) | 8.41 (11) | 6.23 (10) | 2.69 (5) | 3.25 (8) | 2.29 (3) | 2.91 (7) | 2.70 (6) |
| | ResNet50 | 4.35 (9) | 3.66 (8) | 3.43 (7) | 2.88 (3) | 6.58 (11) | 6.04 (10) | **2.45 (1)** | 3.08 (4) | 2.59 (2) | 3.23 (6) | 3.17 (5) |
| | ResNet110 | 3.61 (6) | 2.70 (2) | 4.19 (9) | 4.09 (8) | 7.53 (11) | 5.70 (10) | **2.49 (1)** | 3.04 (3) | 3.30 (4) | 3.99 (7) | 3.44 (5) |
| | ResNet152 | 3.40 (6) | 3.29 (5) | 2.88 (2) | 4.30 (9) | 6.70 (11) | 6.06 (10) | **2.48 (1)** | 2.95 (4) | 2.88 (3) | 3.41 (7) | 3.58 (8) |
| | Avg. gain | — | – 0.62 | – 0.56 | – 0.34 | + 3.59 | + 2.30 | – 1.17 | – 0.62 | – 0.93 | – 0.32 | – 0.48 |
| Tiny-ImageNet | ResNet18 | 1.59 (3) | 1.60 (4) | 1.59 (2) | 1.79 (9) | 2.19 (11) | 1.82 (10) | 1.75 (8) | 1.63 (5) | **1.56 (1)** | 1.73 (7) | 1.70 (6) |
| | ResNet50 | 1.57 (3) | 1.67 (5) | 1.77 (6) | 2.04 (9) | 2.06 (10) | 2.02 (8) | 1.79 (7) | 2.16 (11) | 1.57 (2) | **1.50 (1)** | 1.66 (4) |
| | ResNet110 | 1.67 (4) | 1.94 (8) | 1.75 (6) | 2.63 (11) | 2.33 (10) | **1.47 (1)** | 2.10 (9) | 1.76 (7) | 1.69 (5) | 1.63 (2) | 1.64 (3) |
| | ResNet152 | 1.65 (4) | 2.45 (8) | 2.79 (9) | 2.39 (7) | 3.11 (11) | 2.07 (6) | 1.61 (3) | 2.92 (10) | 1.76 (5) | **1.37 (1)** | 1.55 (2) |
| | ResNet18† | 1.55 (2) | 1.76 (7) | 1.63 (4) | 1.60 (3) | 2.89 (11) | 2.23 (10) | 2.08 (9) | 1.76 (5) | 1.76 (6) | 1.87 (8) | **1.51 (1)** |
| | ResNet152† | 2.53 (6) | 2.23 (5) | 2.71 (8) | 2.64 (7) | 7.05 (11) | 6.25 (10) | **1.70 (1)** | 2.12 (4) | 2.08 (3) | 2.93 (9) | 1.75 (2) |
| | Avg. gain | — | + 0.18 | + 0.27 | + 0.41 | + 1.51 | + 0.88 | + 0.07 | + 0.29 | – 0.02 | + 0.07 | – 0.12 |

Table B. The overall comparative results in terms of the **Calibrated AdaECE**.

| | Backbones | ERM | Mixup (0.1) | Mixup (0.5) | Mixup (1.0) | Mixup (DT) | Mixup (TO) | Mixup (SC) | Mixup (IO) | **MIT-A** | **MIT-L** ($\Delta\lambda{>}\frac{1}{2}$) | **MIT-A** ($\Delta\lambda{>}\frac{1}{2}$) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SVHN | ResNet18 | 0.70 (2) | 1.21 (7) | 1.60 (9) | 1.37 (8) | 1.75 (10) | 1.87 (11) | 0.81 (6) | 0.78 (5) | **0.60 (1)** | 0.71 (3) | 0.78 (4) |
| | ResNet50 | 1.09 (6) | 1.47 (8) | 1.60 (9) | 1.41 (7) | 2.35 (10) | 2.47 (11) | 0.72 (2) | **0.69 (1)** | 0.74 (5) | 0.73 (4) | 0.72 (3) |
| | ResNet110 | 1.03 (6) | 1.52 (8) | 1.77 (9) | 1.32 (7) | 2.42 (11) | 2.25 (10) | 0.77 (3) | 0.80 (4) | **0.69 (1)** | 0.77 (2) | 0.84 (5) |
| | ResNet152 | 1.15 (6) | 1.47 (8) | 1.50 (9) | 1.25 (7) | 2.26 (11) | 2.26 (10) | 0.81 (3) | 0.78 (2) | 0.88 (5) | **0.66 (1)** | 0.84 (4) |
| | Avg. gain | — | + 0.42 | + 0.62 | + 0.34 | + 1.19 | + 1.21 | – 0.21 | – 0.23 | – 0.26 | – 0.27 | – 0.19 |
| CIFAR-10 | ResNet18 | 0.73 (6) | 1.72 (9) | 1.68 (8) | 1.36 (7) | 3.40 (11) | 2.90 (10) | 0.55 (4) | 0.57 (5) | **0.46 (1)** | 0.52 (2) | 0.55 (3) |
| | ResNet50 | 0.65 (5) | 2.04 (9) | 1.92 (8) | 1.24 (7) | 3.85 (11) | 3.33 (10) | 0.51 (3) | **0.39 (1)** | 0.64 (4) | 0.50 (2) | 0.67 (6) |
| | ResNet110 | 0.81 (5) | 2.18 (9) | 1.72 (8) | 1.25 (7) | 3.70 (11) | 3.34 (10) | 0.50 (4) | **0.40 (1)** | 0.45 (2) | 0.48 (3) | 0.87 (6) |
| | ResNet152 | 0.71 (5) | 2.20 (9) | 1.83 (8) | 1.13 (7) | 3.64 (11) | 3.23 (10) | 0.49 (2) | **0.40 (1)** | 0.55 (4) | 0.51 (3) | 0.83 (6) |
| | Avg. gain | — | + 1.30 | + 1.06 | + 0.51 | + 2.92 | + 2.47 | – 0.21 | – 0.28 | – 0.20 | – 0.22 | – 0.01 |
| CIFAR-100 | ResNet18 | 2.54 (9) | 1.68 (5) | **1.08 (1)** | 1.20 (2) | 5.64 (11) | 3.55 (10) | 2.00 (7) | 1.78 (6) | 1.40 (3) | 2.12 (8) | 1.61 (4) |
| | ResNet50 | 2.35 (7) | 1.81 (2) | 2.55 (8) | 2.66 (9) | 5.82 (11) | 5.70 (10) | **1.77 (1)** | 2.01 (5) | 1.85 (4) | 2.07 (6) | 1.83 (3) |
| | ResNet110 | 2.58 (7) | **1.35 (1)** | 3.32 (9) | 3.05 (8) | 6.07 (11) | 5.38 (10) | 1.65 (2) | 1.87 (4) | 1.84 (3) | 2.17 (6) | 1.89 (5) |
| | ResNet152 | 2.34 (6) | 1.77 (3) | 2.57 (8) | 3.63 (9) | 5.56 (11) | 5.05 (10) | 1.58 (2) | 1.88 (4) | **1.57 (1)** | 2.36 (7) | 2.00 (5) |
| | Avg. gain | — | – 0.80 | – 0.07 | + 0.18 | + 3.32 | + 2.46 | – 0.70 | – 0.56 | – 0.78 | – 0.27 | – 0.62 |
| Tiny-ImageNet | ResNet18 | 1.40 (4) | **1.32 (1)** | 1.34 (2) | 1.47 (5) | 15.6 (9) | 17.0 (11) | 16.0 (10) | 15.4 (8) | 1.50 (6) | 1.57 (7) | 1.38 (3) |
| | ResNet18† | 1.17 (2) | 1.30 (4) | 1.63 (6) | 1.69 (7) | 16.0 (10) | 15.6 (8) | 15.7 (9) | 16.0 (11) | 1.22 (3) | **1.17 (1)** | 1.40 (5) |
| | ResNet110 | **1.31 (1)** | 1.49 (5) | 1.64 (6) | 2.20 (7) | 16.2 (9) | 16.6 (10) | 16.8 (11) | 15.1 (8) | 1.36 (2) | 1.38 (3) | 1.38 (4) |
| | ResNet152 | 1.35 (2) | 2.20 (7) | 1.88 (5) | 2.04 (6) | 15.6 (10) | 15.4 (9) | 16.3 (11) | 10.9 (8) | 1.51 (4) | **1.26 (1)** | 1.36 (3) |
| | ResNet18 | 1.14 (2) | 1.46 (6) | 1.16 (3) | 1.29 (4) | 2.82 (11) | 1.90 (10) | 1.47 (7) | 1.65 (8) | 1.45 (5) | 1.67 (9) | **1.04 (1)** |
| | ResNet152† | 1.80 (6) | 1.76 (5) | 2.66 (8) | 2.67 (9) | 5.23 (10) | 7.12 (11) | 1.25 (2) | 1.69 (4) | 1.28 (3) | 2.49 (7) | **1.18 (1)** |
| | Avg. gain | — | + 0.22 | + 0.35 | + 0.53 | + 10.5 | + 10.9 | + 9.92 | + 8.78 | + 0.02 | + 0.22 | – 0.07 |

Table C. The overall comparative results in terms of the **Calibrated TECE**.

| | Backbones | ERM | Mixup (0.1) | Mixup (0.5) | Mixup (1.0) | Mixup (DT) | Mixup (TO) | Mixup (SC) | Mixup (IO) | **MIT-A** | **MIT-L** ($\Delta\lambda{>}\frac{1}{2}$) | **MIT-A** ($\Delta\lambda{>}\frac{1}{2}$) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SVHN | ResNet18 | **0.37 (1)** | 0.89 (7) | 1.04 (11) | 0.93 (9) | 0.90 (8) | 0.98 (10) | 0.38 (3) | 0.49 (5) | 0.37 (2) | 0.54 (6) | 0.43 (4) |
| | ResNet50 | 0.79 (6) | 2.20 (11) | 1.08 (8) | 1.04 (7) | 1.44 (10) | 1.28 (9) | 0.47 (4) | **0.38 (1)** | 0.41 (2) | 0.41 (3) | 0.56 (5) |
| | ResNet110 | 0.65 (6) | 0.96 (7) | 1.18 (9) | 1.02 (8) | 1.93 (11) | 1.65 (10) | 0.40 (2) | 0.52 (4) | **0.40 (1)** | 0.41 (3) | 0.61 (5) |
| | ResNet152 | 0.80 (6) | 0.97 (8) | 1.11 (9) | 0.94 (7) | 1.17 (10) | 1.26 (11) | 0.48 (2) | 0.63 (5) | 0.54 (3) | **0.44 (1)** | 0.61 (4) |
| | Avg. gain | — | + 0.60 | + 0.45 | + 0.33 | + 0.70 | + 0.64 | – 0.21 | – 0.14 | – 0.22 | – 0.20 | – 0.09 |
| CIFAR-10 | ResNet18 | 0.49 (6) | 0.91 (8) | 1.04 (9) | 0.82 (7) | 1.57 (11) | 1.31 (10) | 0.47 (4) | **0.42 (1)** | 0.43 (2) | 0.45 (3) | 0.49 (5) |
| | ResNet50 | 0.63 (6) | 0.96 (8) | 1.02 (9) | 0.80 (7) | 3.43 (11) | 1.53 (10) | 0.49 (3) | **0.31 (1)** | 0.51 (4) | 0.43 (2) | 0.54 (5) |
| | ResNet110 | 0.66 (5) | 0.98 (9) | 0.85 (8) | 0.74 (7) | 1.36 (10) | 2.49 (11) | 0.41 (3) | **0.37 (1)** | 0.40 (2) | 0.43 (4) | 0.66 (6) |
| | ResNet152 | 0.54 (6) | 1.02 (9) | 0.97 (8) | 0.70 (7) | 1.40 (10) | 1.72 (11) | 0.49 (4) | **0.34 (1)** | 0.46 (3) | 0.35 (2) | 0.53 (5) |
| | Avg. gain | — | + 0.38 | + 0.39 | + 0.18 | + 1.36 | + 1.18 | – 0.11 | – 0.21 | – 0.12 | – 0.16 | – 0.02 |
| CIFAR-100 | ResNet18 | 1.46 (8) | **0.84 (1)** | 0.97 (5) | 0.85 (3) | 2.31 (11) | 1.79 (10) | 1.16 (7) | 0.85 (2) | 1.02 (6) | 1.52 (9) | 0.91 (4) |
| | ResNet50 | 1.41 (7) | 1.22 (4) | 2.22 (9) | 2.08 (8) | 4.29 (10) | 4.85 (11) | 1.09 (2) | 1.15 (3) | 1.26 (5) | 1.39 (6) | **0.99 (1)** |
| | ResNet110 | 1.37 (7) | **0.87 (1)** | 2.75 (9) | 2.18 (8) | 6.95 (11) | 4.86 (10) | 1.12 (5) | 0.90 (2) | 1.00 (3) | 1.37 (6) | 1.01 (4) |
| | ResNet152 | 1.48 (7) | 1.11 (5) | 2.27 (8) | 2.47 (9) | 3.41 (10) | 5.75 (11) | 1.10 (4) | **1.01 (1)** | 1.04 (2) | 1.33 (6) | 1.09 (3) |
| | Avg. gain | — | – 0.41 | + 0.62 | + 0.46 | + 2.80 | + 2.88 | – 0.31 | – 0.45 | – 0.35 | – 0.02 | – 0.42 |
| Tiny-ImageNet | ResNet18 | 1.46 (3) | 2.28 (10) | 1.68 (5) | 1.77 (8) | 2.42 (11) | 1.75 (7) | 1.70 (6) | 1.79 (9) | 1.47 (4) | 1.20 (2) | **1.16 (1)** |
| | ResNet50 | 1.31 (8) | **0.82 (1)** | 2.03 (10) | 1.20 (6) | 2.29 (11) | 1.48 (9) | 1.16 (5) | 1.25 (7) | 1.12 (4) | 0.89 (2) | 1.02 (3) |
| | ResNet110 | 1.26 (4) | 2.70 (9) | 3.12 (10) | 8.88 (11) | 2.23 (7) | 1.51 (5) | 2.37 (8) | 1.66 (6) | 1.07 (2) | 1.11 (3) | **1.06 (1)** |
| | ResNet152 | 1.27 (2) | 1.49 (4) | 1.31 (3) | 5.31 (10) | 4.88 (9) | 1.95 (7) | 1.56 (5) | 12.1 (11) | 2.23 (8) | **1.19 (1)** | 1.82 (6) |
| | ResNet18† | 1.04 (4) | 1.06 (5) | **0.87 (1)** | 1.33 (8) | 1.46 (10) | 1.16 (7) | 2.22 (11) | 0.94 (3) | 0.94 (2) | 1.38 (9) | 1.12 (6) |
| | ResNet152† | 1.40 (6) | 0.93 (4) | 2.69 (8) | 2.83 (9) | 5.31 (10) | 9.67 (11) | 0.81 (2) | 1.59 (7) | 0.85 (3) | 1.36 (5) | **0.68 (1)** |
| | Avg. gain | — | + 0.25 | + 0.66 | + 2.26 | + 1.80 | + 1.63 | + 0.34 | + 1.93 | – 0.01 | – 0.10 | – 0.14 |

Table D. The overall comparative results in terms of the **Calibrated NLL**.

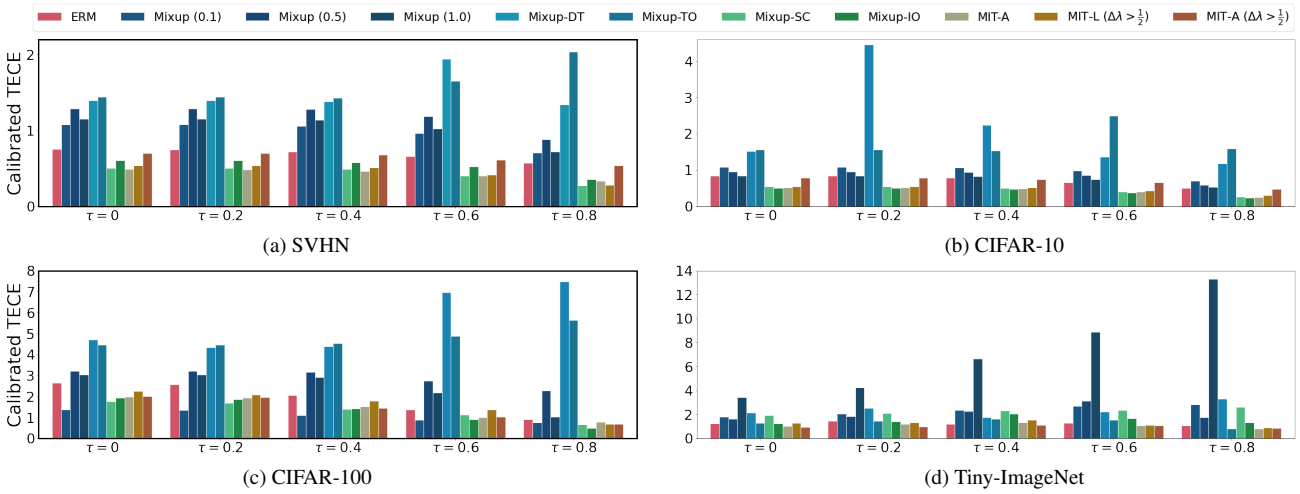
| | Backbones | ERM | Mixup (0.1) | Mixup (0.5) | Mixup (1.0) | Mixup (DT) | Mixup (TO) | Mixup (SC) | Mixup (IO) | **MIT-A** | **MIT-L** ($\Delta\lambda{>}\frac{1}{2}$) | **MIT-A** ($\Delta\lambda{>}\frac{1}{2}$) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SVHN | ResNet18 | 0.16 (2) | 0.17 (3) | 0.20 (8) | 0.22 (9) | 0.18 (6) | 0.17 (4) | 0.20 (7) | 0.23 (10) | 0.18 (5) | 0.25 (11) | **0.15 (1)** |
| | ResNet50 | 0.15 (3) | 0.15 (4) | 0.17 (6) | 0.19 (8) | 0.20 (9) | 0.21 (10) | 0.17 (5) | 0.19 (7) | 0.14 (2) | 0.22 (11) | **0.14 (1)** |
| | ResNet110 | 0.15 (3) | 0.15 (4) | 0.16 (5) | 0.18 (8) | 0.21 (11) | 0.20 (10) | 0.16 (6) | 0.17 (7) | 0.13 (2) | 0.19 (9) | **0.12 (1)** |
| | ResNet152 | 0.14 (4) | 0.14 (3) | 0.15 (5) | 0.17 (7) | 0.21 (11) | 0.19 (9) | 0.16 (6) | 0.17 (8) | 0.13 (2) | 0.19 (10) | **0.12 (1)** |
| | Avg. gain | — | + 0.01 | + 0.01 | + 0.03 | + 0.04 | + 0.04 | + 0.01 | + 0.03 | – 0.01 | + 0.06 | – 0.01 |
| CIFAR-10 | ResNet18 | 0.17 (9) | 0.17 (8) | 0.16 (4) | 0.16 (5) | 0.25 (11) | 0.23 (10) | 0.17 (7) | 0.16 (6) | 0.13 (2) | 0.15 (3) | **0.12 (1)** |
| | ResNet50 | 0.18 (9) | 0.18 (8) | 0.16 (7) | 0.16 (5) | 0.31 (11) | 0.25 (10) | 0.16 (6) | 0.15 (4) | 0.13 (2) | 0.13 (3) | **0.12 (1)** |
| | ResNet110 | 0.17 (9) | 0.16 (8) | 0.15 (6) | 0.16 (7) | 0.29 (11) | 0.26 (10) | 0.15 (5) | 0.14 (4) | **0.12 (1)** | 0.12 (3) | 0.12 (2) |
| | ResNet152 | 0.16 (8) | 0.16 (9) | 0.15 (7) | 0.14 (5) | 0.28 (11) | 0.24 (10) | 0.15 (6) | 0.13 (4) | 0.11 (2) | 0.12 (3) | **0.11 (1)** |
| | Avg. gain | — | – 0.00 | – 0.01 | – 0.01 | + 0.11 | + 0.07 | – 0.01 | – 0.02 | – 0.04 | – 0.03 | – 0.04 |
| CIFAR-100 | ResNet18 | 0.97 (7) | 0.96 (6) | 0.92 (5) | 0.90 (4) | 1.28 (11) | 1.04 (10) | 1.02 (8) | 1.04 (9) | 0.84 (2) | 0.88 (3) | **0.84 (1)** |
| | ResNet50 | 0.98 (8) | 0.91 (6) | 0.89 (4) | 0.90 (5) | 1.51 (11) | 1.20 (10) | 1.00 (9) | 0.95 (7) | **0.77 (1)** | 0.84 (3) | 0.80 (2) |
| | ResNet110 | 0.89 (8) | 0.85 (6) | 0.82 (4) | 0.85 (5) | 1.43 (11) | 1.10 (10) | 0.92 (9) | 0.86 (7) | 0.76 (2) | 0.80 (3) | **0.74 (1)** |
| | ResNet152 | 0.92 (9) | 0.84 (5) | 0.82 (4) | 0.85 (6) | 1.32 (11) | 1.12 (10) | 0.91 (8) | 0.87 (7) | 0.74 (2) | 0.78 (3) | **0.72 (1)** |
| | Avg. gain | — | – 0.04 | – 0.07 | – 0.06 | + 0.44 | + 0.17 | + 0.02 | – 0.01 | – 0.16 | – 0.11 | – 0.16 |
| Tiny-ImageNet | ResNet18 | 2.36 (6) | 2.39 (8) | 2.37 (7) | 2.33 (5) | 3.14 (11) | 2.51 (10) | 2.48 (9) | 2.31 (4) | **2.13 (1)** | 2.21 (3) | 2.15 (2) |
| | ResNet50 | 2.22 (5) | 2.24 (8) | 2.23 (7) | 2.22 (6) | 3.10 (11) | 2.41 (10) | 2.34 (9) | 2.20 (4) | 2.04 (2) | 2.09 (3) | **2.03 (1)** |
| | ResNet110 | 2.28 (3) | 2.62 (8) | 2.59 (7) | 3.44 (11) | 3.19 (10) | 2.66 (9) | 2.52 (6) | 2.46 (5) | 2.19 (2) | 2.43 (4) | **2.07 (1)** |
| | ResNet152 | 2.38 (3) | 3.03 (7) | 3.63 (9) | 3.76 (10) | 3.36 (8) | 2.64 (6) | 2.59 (5) | 4.55 (11) | 2.32 (2) | 2.49 (4) | **2.26 (1)** |
| | ResNet18† | 1.99 (4) | 2.05 (7) | 2.03 (5) | 2.05 (6) | 2.71 (11) | 2.11 (8) | 2.16 (10) | 2.14 (9) | 1.90 (2) | 1.93 (3) | **1.88 (1)** |
| | ResNet152† | 1.60 (4) | 1.59 (2) | 1.64 (6) | 1.69 (7) | 2.55 (11) | 1.88 (10) | 1.74 (9) | 1.74 (8) | **1.56 (1)** | 1.59 (3) | 1.60 (5) |
| | Avg. gain | — | + 0.18 | + 0.27 | + 0.44 | + 0.86 | + 0.23 | + 0.16 | + 0.42 | – 0.11 | – 0.01 | – 0.13 |

Table E. The overall comparative results in terms of the Accuracy, Calibrated ECE and Calibrated NLL on **DenseNet-121**.

| | Metrics | ERM | Mixup (0.1) | Mixup (0.5) | Mixup (1.0) | Mixup (DT) | Mixup (TO) | Mixup (SC) | Mixup (IO) | **MIT-A** | **MIT-L** ($\Delta\lambda>\frac{1}{2}$) | **MIT-A** ($\Delta\lambda>\frac{1}{2}$) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SVHN | Accuracy | 94.9 (9) | 95.8 (7) | 96.5 (2) | **96.5 (1)** | 94.1 (11) | 94.6 (10) | 95.4 (8) | 95.8 (6) | 96.1 (4) | 96.4 (3) | 96.1 (5) |
| | ECE | 0.54 (2) | 1.00 (7) | 0.93 (6) | 1.18 (9) | 1.37 (11) | 1.21 (10) | 1.02 (8) | **0.48 (1)** | 0.66 (3) | 0.71 (4) | 0.85 (5) |
| | NLL | 0.14 (3) | 0.16 (5) | 0.17 (6) | 0.17 (8) | 0.23 (11) | 0.21 (10) | 0.16 (4) | 0.17 (7) | 0.19 (9) | **0.12 (1)** | 0.13 (2) |
| CIFAR-10 | Accuracy | 94.9 (9) | 95.8 (7) | 96.5 (2) | **96.5 (1)** | 94.1 (11) | 94.6 (10) | 95.4 (8) | 95.8 (6) | 96.1 (4) | 96.4 (3) | 96.1 (5) |
| | ECE | 1.08 (8) | 1.13 (9) | 0.95 (7) | 0.91 (6) | 1.46 (10) | 1.48 (11) | 0.48 (2) | 0.57 (4) | 0.53 (3) | 0.76 (5) | **0.38 (1)** |
| | NLL | 0.16 (9) | 0.16 (8) | 0.15 (6) | 0.15 (7) | 0.28 (11) | 0.25 (10) | 0.14 (5) | 0.13 (4) | 0.12 (3) | **0.11 (1)** | 0.11 (2) |
| CIFAR-100 | Accuracy | 77.4 (8) | 78.5 (6) | 80.4 (3) | 80.5 (2) | 73.9 (11) | 77.7 (7) | 76.4 (10) | 76.8 (9) | 79.0 (4) | **80.6 (1)** | 78.8 (5) |
| | ECE | 2.61 (7) | 1.35 (2) | 3.59 (9) | 3.10 (8) | 5.41 (11) | 4.22 (10) | 1.96 (4) | 2.02 (5) | 2.07 (6) | 1.88 (3) | **1.29 (1)** |
| | NLL | 0.85 (7) | 0.82 (6) | 0.79 (4) | 0.79 (5) | 1.34 (11) | 1.16 (10) | 0.86 (8) | 0.87 (9) | 0.75 (3) | **0.69 (1)** | 0.74 (2) |

Table F. The overall comparative results in terms of the Calibrated ECE with **Histogram Binning**.

| | Metrics | ERM | Mixup (0.1) | Mixup (0.5) | Mixup (1.0) | Mixup (DT) | Mixup (TO) | Mixup (SC) | Mixup (IO) | **MIT-A** | **MIT-L** ($\Delta\lambda>\frac{1}{2}$) | **MIT-A** ($\Delta\lambda>\frac{1}{2}$) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SVHN | ResNet18 | **0.61 (1)** | 0.88 (7) | 1.10 (10) | 4.34 (11) | 0.92 (9) | 0.72 (4) | 0.76 (6) | 0.67 (2) | 0.90 (8) | 0.73 (5) | 0.70 (3) |
| | ResNet50 | 0.77 (10) | 0.71 (6) | 0.68 (3) | 0.97 (11) | 0.71 (5) | 0.75 (8) | 0.66 (2) | 0.77 (9) | **0.63 (1)** | 0.74 (7) | 0.70 (4) |
| CIFAR-10 | ResNet18 | 0.56 (3) | 0.63 (6) | 2.11 (11) | 0.85 (10) | 0.80 (9) | 0.78 (8) | **0.50 (1)** | 0.65 (7) | 0.62 (5) | 0.51 (2) | 0.56 (4) |
| | ResNet50 | 0.74 (8) | 0.59 (5) | 2.00 (10) | 6.14 (11) | 0.77 (9) | 0.71 (7) | 0.65 (6) | 0.52 (3) | 0.55 (4) | 0.50 (2) | **0.49 (1)** |
| CIFAR-100 | ResNet18 | 1.29 (2) | 1.47 (9) | 1.31 (3) | 1.36 (6) | 1.35 (4) | 2.31 (11) | 1.46 (8) | 1.48 (10) | **1.07 (1)** | 1.38 (7) | 1.35 (5) |
| | ResNet50 | **1.02 (1)** | 1.40 (9) | 1.31 (6) | 1.38 (7) | 1.46 (10) | 1.39 (8) | 1.25 (5) | 1.24 (4) | 1.11 (2) | 1.16 (3) | 1.51 (11) |




(a) SVHN (b) CIFAR-10

(c) CIFAR-100 (d) Tiny-ImageNet

Figure B. The comparative results in terms of the **Calibrated TECE** with different choices of threshold $\tau$ on ResNet110.

consistent to that of ResNets. Table F shows the results of Histogram Binning (HB) and Beta Calibration (BC). It is shown that mixup's issue can also be mitigated by our methods in these cases.

## 4.2. Comparison with other Calibration Methods

In this subsection, we compare our approaches with three commonly considered calibration baselines Brier loss [1], label smoothing [8], Focal loss [7], as well as two specifically designed calibration loss Maximum Mean Calibration Error (MMCE) [5] and Difference between Confidence and Accuracy (DCA) [6]. We briefly introduce these methods as follows:

- Brier loss is the simple the mean squared error between the predicted confidences and the ground-truth one-hot labels, which was considered as an important baseline as it can be decomposed into calibration and refinement [1].

Table G. The comparison with other methods on **calibrated ECE, NLL, ACE, AdaECE and TECE** (with ResNet18).

| Methods | SVHN | | | | | | CIFAR-10 | | | | | | CIFAR-100 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc | ECE | ACE | NLL | AdaECE | TECE | Acc | ECE | ACE | NLL | AdaECE | TECE | Acc | ECE | ACE | NLL | AdaECE | TECE |
| ERM | 95.4 | 0.50 | 5.74 | **0.16** | 0.70 | 0.37 | 94.5 | 0.65 | 4.81 | 0.17 | 0.73 | 0.49 | 74.4 | 2.56 | 3.46 | 0.97 | 2.54 | 1.46 |
| Brier | 95.5 | 0.60 | 6.01 | 0.16 | **0.53** | 0.45 | 94.3 | 1.06 | 5.07 | 0.19 | 1.10 | 0.86 | 74.4 | 3.30 | 4.44 | 1.00 | 3.16 | 1.45 |
| LS ($\epsilon = 0.01$) | 95.7 | 0.78 | 6.47 | 0.16 | 1.40 | 0.70 | 94.5 | 1.35 | 5.59 | 0.19 | 2.30 | 1.31 | 73.7 | 2.92 | 4.13 | 1.08 | 2.77 | 1.52 |
| LS ($\epsilon = 0.05$) | 95.6 | 0.93 | 5.64 | 0.17 | 1.68 | 0.84 | 94.7 | 1.39 | 5.97 | 0.20 | 2.78 | 1.21 | 75.5 | 3.52 | 4.92 | 1.06 | 3.55 | 1.24 |
| LS ($\epsilon = 0.1$) | **95.8** | 0.85 | 5.52 | 0.16 | 1.53 | 0.73 | 94.7 | 1.39 | 5.64 | 0.21 | 2.87 | 1.21 | 76.1 | 3.38 | 6.10 | 1.04 | 3.52 | 1.27 |
| Focal Loss ($\gamma = 1.0$) | 95.6 | **0.44** | **4.75** | 0.16 | 0.92 | 0.37 | 94.3 | 0.97 | **4.04** | 0.18 | 0.84 | 0.72 | 73.9 | 1.82 | 2.29 | 0.94 | 1.71 | 1.25 |
| Focal Loss ($\gamma = 2.0$) | 95.6 | 0.57 | 7.72 | 0.16 | 0.76 | 0.43 | 94.4 | 0.89 | 5.42 | 0.17 | 1.14 | 0.75 | 74.1 | 1.26 | **1.88** | 0.92 | 1.05 | **0.73** |
| Focal Loss ($\gamma = 3.0$) | 95.6 | 0.69 | 6.79 | 0.16 | 0.75 | 0.59 | 93.7 | 1.23 | 5.84 | 0.19 | 1.46 | 1.02 | 74.0 | **1.04** | 3.66 | 0.91 | **0.98** | 0.75 |
| MMCE ($\beta = 0.1$) | 95.5 | 0.67 | 5.38 | 0.16 | 0.75 | 0.45 | 94.4 | 0.63 | 7.28 | 0.17 | 0.50 | 0.44 | 73.8 | 2.59 | 3.84 | 0.99 | 2.49 | 1.56 |
| MMCE ($\beta = 0.5$) | 95.1 | 1.15 | 6.44 | 0.18 | 0.99 | 0.78 | 94.2 | 1.05 | 8.62 | 0.19 | 0.94 | 0.63 | 72.2 | 2.04 | 2.95 | 1.03 | 1.97 | 1.30 |
| MMCE ($\beta = 1.0$) | 94.9 | 1.41 | 8.21 | 0.21 | 1.44 | 0.92 | 94.0 | 3.29 | 13.6 | 0.26 | 2.86 | 1.14 | 71.7 | 1.80 | 2.52 | 1.05 | 1.81 | 1.29 |
| DCA ($\eta = 0.1$) | 95.1 | 0.50 | 5.60 | 0.17 | 0.87 | **0.33** | 94.5 | 0.87 | 5.57 | 0.17 | 0.59 | 0.71 | 74.5 | 2.72 | 3.65 | 0.97 | 2.65 | 1.64 |
| DCA ($\eta = 0.5$) | 95.5 | 0.66 | 5.20 | 0.16 | 0.76 | 0.43 | 94.4 | 0.66 | 4.75 | 0.17 | 0.65 | 0.49 | 73.6 | 2.93 | 4.56 | 1.01 | 2.79 | 1.58 |
| DCA ($\eta = 1.0$) | 95.5 | 0.62 | 5.44 | 0.16 | 0.73 | 0.49 | 94.5 | 0.71 | 6.46 | 0.18 | 0.57 | 0.55 | 73.3 | 2.83 | 4.07 | 1.01 | 2.90 | 1.66 |
| MIT-A | 95.0 | 0.47 | 5.00 | 0.18 | 0.60 | 0.37 | **95.5** | **0.56** | 6.02 | **0.13** | **0.46** | **0.43** | **76.2** | 1.44 | 2.29 | **0.84** | 1.40 | 1.02 |

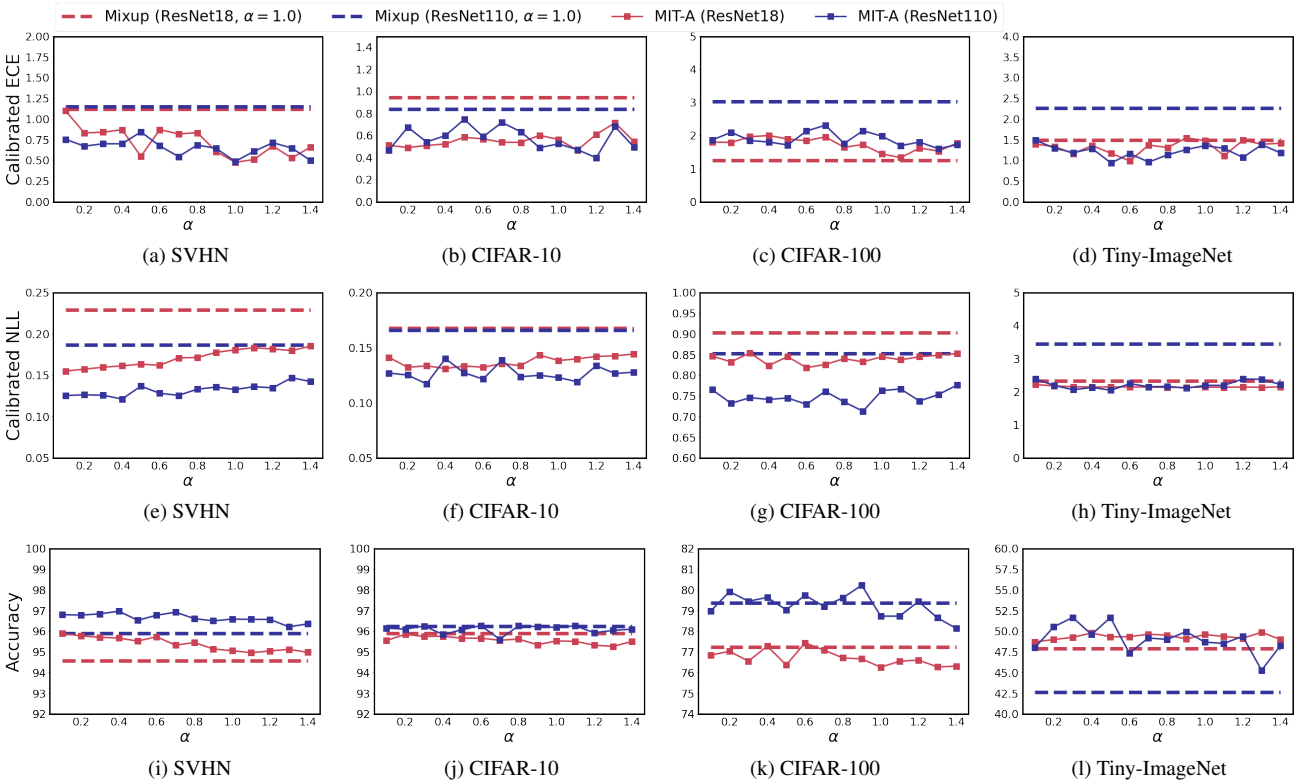
Table H. The comparison with other methods on **calibrated ECE, NLL, ACE, AdaECE and TECE** (with ResNet110).

| Methods | SVHN | | | | | | CIFAR-10 | | | | | | CIFAR-100 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc | ECE | ACE | NLL | AdaECE | TECE | Acc | ECE | ACE | NLL | AdaECE | TECE | Acc | ECE | ACE | NLL | AdaECE | TECE |
| ERM | 96.0 | 0.75 | 6.77 | 0.15 | 1.03 | 0.65 | 94.7 | 0.83 | 6.72 | 0.17 | 0.81 | 0.66 | 76.1 | 2.64 | 3.61 | 0.89 | 2.58 | 1.37 |
| Brier | 95.8 | 1.08 | 14.8 | 0.18 | 1.58 | 1.04 | 94.4 | 0.72 | 5.13 | 0.18 | 0.71 | 0.56 | 73.1 | 2.75 | 5.44 | 1.03 | 2.78 | 1.54 |
| LS ($\epsilon = 0.01$) | 96.1 | 1.10 | 9.94 | 0.16 | 1.85 | 1.03 | 94.7 | 1.31 | 6.26 | 0.22 | 2.79 | 1.19 | 76.1 | 2.61 | 4.16 | 0.99 | 2.55 | 1.37 |
| LS ($\epsilon = 0.05$) | 95.8 | 1.30 | 7.03 | 0.19 | 2.30 | 1.29 | 94.8 | 1.43 | 13.0 | 0.23 | 2.95 | 1.28 | 75.7 | 3.32 | 5.33 | 1.07 | 3.58 | 2.14 |
| LS ($\epsilon = 0.1$) | 95.9 | 1.27 | 5.88 | 0.19 | 2.19 | 1.16 | 94.9 | 1.28 | 22.0 | 0.23 | 2.97 | 1.20 | 76.7 | 3.38 | 5.42 | 1.07 | 4.25 | 3.07 |
| Focal Loss ($\gamma = 1.0$) | 95.9 | 0.84 | 4.15 | 0.15 | 1.07 | 0.76 | 94.6 | 0.71 | 6.94 | 0.17 | 0.77 | 0.62 | 76.1 | 1.50 | 2.47 | 0.85 | 1.49 | 0.84 |
| Focal Loss ($\gamma = 2.0$) | 96.0 | 0.98 | 5.71 | 0.15 | 1.43 | 0.86 | 94.2 | 1.03 | 5.52 | 0.18 | 1.15 | 0.91 | 75.8 | **0.92** | 1.87 | 0.85 | **1.01** | **0.69** |
| Focal Loss ($\gamma = 3.0$) | 96.0 | 0.98 | 4.56 | 0.15 | 1.49 | 0.93 | 93.8 | 1.22 | 5.04 | 0.19 | 1.32 | 1.07 | 76.0 | 1.24 | **1.83** | 0.84 | 1.05 | 1.04 |
| MMCE ($\beta = 0.1$) | 95.9 | 0.63 | 4.50 | 0.15 | 0.96 | 0.55 | 94.7 | 0.59 | 6.36 | 0.16 | 0.51 | 0.40 | 74.7 | 2.36 | 4.23 | 0.94 | 2.24 | 1.25 |
| MMCE ($\beta = 0.5$) | 95.8 | 1.37 | 14.5 | 0.18 | 1.54 | **0.38** | 94.6 | 1.67 | 12.5 | 0.19 | 1.73 | **0.38** | 75.0 | 1.91 | 2.72 | 0.91 | 1.93 | 1.30 |
| MMCE ($\beta = 1.0$) | 95.8 | 1.72 | 16.3 | 0.19 | 1.60 | 0.42 | 93.3 | 4.17 | 15.5 | 0.28 | 2.99 | 0.73 | 74.6 | 1.96 | 5.05 | 0.92 | 1.68 | 1.16 |
| DCA ($\eta = 0.1$) | 96.1 | 0.79 | 6.55 | 0.15 | 1.02 | 0.71 | 95.0 | 0.71 | 5.00 | 0.16 | 0.58 | 0.55 | 75.9 | 2.63 | 3.64 | 0.90 | 2.71 | 1.21 |
| DCA ($\eta = 0.5$) | 96.0 | 0.86 | **3.53** | 0.15 | 1.06 | 0.78 | 94.7 | 0.58 | **4.51** | 0.17 | 0.75 | 0.52 | 76.0 | 2.38 | 3.60 | 0.89 | 2.34 | 1.18 |
| DCA ($\eta = 1.0$) | 95.9 | 0.86 | 4.98 | 0.15 | 0.89 | 0.74 | 94.8 | 0.66 | 5.27 | 0.17 | 0.61 | 0.53 | 76.0 | 2.65 | 3.61 | 0.89 | 2.45 | 1.31 |
| MIT-A | **96.5** | **0.48** | 6.29 | **0.13** | **0.69** | 0.40 | **96.1** | **0.52** | 8.18 | **0.12** | **0.45** | 0.40 | **78.7** | 1.98 | 3.30 | **0.76** | 1.84 | 1.00 |

- Label smoothing (LS) is widely used to reduce overfitting of DNNs [8]. The mechanism of LS is simple: when training with CE loss, the one-hot label vector $y$ is replaced with *soft* label vector $\widetilde{y}$, whose elements can be formally denoted as $\widetilde{y_i} = (1-\epsilon)y_i + \epsilon/K, \forall i \in \{1, ..., K\}$, where $\epsilon > 0$ is a strength coefficient.
- Focal loss is originally proposed to address the class imbalance problem in object detection. Formally, for classification tasks where the target distribution is one-hot encoding, it is defined as: $\mathcal{L}_f = -(1-f_y^\theta)^\gamma \log f_y^\theta$, where $\gamma$ is a predefined coefficient. Mukhoti *et al.* [7] found that the models learned by focal loss produce output probabilities which are already very well calibrated.
- MMCE is a continuous and differentiable proxy for calibration error and is normally used as a regularizer alongside the commonly used cross-entropy loss, where a weighting factor $\beta$ could be used to balance the contribution of MMCE [5].
- DCA directly computes the absolute difference between the average confidence and accuracy of training data. It can be also used as an auxiliary loss term alongside the cross-entropy and weighted by a factor $\eta$ [6].

Table G and H show the results of the above comparison methods with different hyperparameters. We can observe similar phenomenon with the comparison between our approaches with mixup and its variants: These methods improve calibration on raw outputs compared with ERM, but often degrade the calibration performance after post-hoc processing. We think this

may be also caused by the confidence penalty effect implicitly induced by these methods. On the contrary, our approaches do not penalize the confidence in training and hence yield bad calibration on the raw outputs, while achieving impressive calibration performance after post-calibration.




(a) SVHN (b) CIFAR-10 (c) CIFAR-100 (d) Tiny-ImageNet

(e) SVHN (f) CIFAR-10 (g) CIFAR-100 (h) Tiny-ImageNet

(i) SVHN (j) CIFAR-10 (k) CIFAR-100 (l) Tiny-ImageNet

Figure C. The results of MIT-A on **Calibrated ECE, Calibrated NLL** and **Accuracy** with different $\alpha$ on ResNet18 and ResNet110.

### 4.3. Impact of the Hyperparameter

We also conduct the empirical study on the mixing factor $\alpha$. Figure C shows the results of MIT-A with $\alpha \in \{0.1, 0.2, 0.3, ..., 1.4\}$. It is illustrated that the calibrated ECE and NLL are not sensitive to the choice of $\alpha$: MIT-A outperforms vanilla mixup in all cases except for the results on CIFAR-100 with ResNet18 (see in Figure C(c)). Moreover, in terms of the accuracy, the performance of our approach is also stable, while can be further improved by specifically selecting an $\alpha$. For example, compared with our default setting ($\alpha = 1$ used in the main experiments), on SVHN/CIFAR-100 with ResNet18/ResNet110, nearly 1%/1.5% improvement of accuracy can be achieved by choosing a relatively smaller $\alpha$.

### 4.4. Results with other Settings

## References


[1] Morris H DeGroot and Stephen E Fienberg. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 32(1-2):12–22, 1983. 7

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 4

[3] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009. 2

[4] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 4

[5] Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In *International Conference on Machine Learning*, pages 2805–2814, 2018. 7, 8

[6] Gongbo Liang, Yu Zhang, Xiaoqin Wang, and Nathan Jacobs. Improved trainable calibration method for neural networks. In *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*. BMVA Press, 2020. 7, 8

[7] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating deep neural networks using focal loss. In *Advances in Neural Information Processing Systems*, pages 15288–15299, 2020. 7, 8

[8] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, pages 4696–4705, 2019. 7, 8

[9] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. 1

[10] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 4

[11] Lukas Neumann, Andrew Zisserman, and Andrea Vedaldi. Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection. *Advances in neural information processing systems Workshop: Machine Learning for Intelligent Transportation Systems*, 2018. 1

[12] Jeremy Nixon, Michael W. Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019. 1, 2

[13] Tianyu Pang, Kun Xu, and Jun Zhu. Mixup inference: Better exploiting mixup to defend adversarial attacks. In *International Conference on Learning Representations*, 2019. 2

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: an imperative style, high-performance deep learning library. In *Neural Information Processing Systems*, volume 33, pages 8026–8037, 2019. 4