Balancing Two Classifiers via A Simplex ETF Structure for Model Calibration

Supplementary Material

A. The proof of Theorem 1

As shown in Definition 1, within the general Simplex ETF framework, in addition to β, the term $\sqrt{\frac{K}{K-1}}$ in the matrix $\mathbf{M}$ serves as a scaling factor, ensuring that each column vector has a fixed length. This scaling ensures that each column vector in $\mathbf{M}$ has an equal norm, which is crucial for maintaining the isometric property required by a Simplex ETF. The scaling factor $\beta\sqrt{\frac{K}{K-1}}$, plays a crucial role in shaping the model's probability distribution, output confidence, and ultimately its classification performance. A detailed analysis of this factor provides a deeper understanding of its adjustment mechanism and core functionality in classification models.

Class probability expression in softmax function. The softmax function is used to map the raw scores to class probabilities. The probability $p_{i,k}$ for class k is given by:

$$p_{i,k} = \frac{\exp\left(\beta\sqrt{\frac{K}{K-1}}\boldsymbol{z}_i\left[\mathbf{U}\left(\mathbf{I}-\frac{1}{K}\mathbf{1}_K\mathbf{1}_K^{\mathrm{T}}\right)\right]_k\right)}{\sum_{j=1}^{K}\exp\left(\beta\sqrt{\frac{K}{K-1}}\boldsymbol{z}_i\left[\mathbf{U}\left(\mathbf{I}-\frac{1}{K}\mathbf{1}_K\mathbf{1}_K^{\mathrm{T}}\right)\right]_j\right)}. \tag{1}$$

To simplify the notation, we define $\sigma_i = \boldsymbol{z}_i\hat{\boldsymbol{m}}_i$, where $\hat{\boldsymbol{m}}_i$ is the i-th column vector of the matrix $\mathbf{U}\left(\mathbf{I}-\frac{1}{K}\mathbf{1}_K\mathbf{1}_K^{\top}\right)$. The class probability expression then simplifies to:

$$p_{i,k} = \frac{\exp\left(\beta\sqrt{\frac{K}{K-1}}\sigma_i\right)}{\sum_{j=1}^{K}\exp\left(\beta\sqrt{\frac{K}{K-1}}\sigma_j\right)}. \tag{2}$$

Further neglecting the normalization constant $\sum_{j=1}^{K}\exp\left(\beta\sqrt{\frac{K}{K-1}}\sigma_j\right)$, the predicted confidence $\hat{p}_i$ is defined as $\hat{p}_i = \max_k p_{i,k}$, which can be approximated as:

$$\hat{p}_i \propto \exp\left(\beta\sqrt{\frac{K}{K-1}}\sigma_i\right). \tag{3}$$

This equation clearly demonstrates the critical role of the scaling factor $\beta\sqrt{\frac{K}{K-1}}$ in the Softmax function. The magnitude of the scaling factor directly determines the concentration of the class probability distribution: a larger $\beta\sqrt{\frac{K}{K-1}}$ significantly increases the probability of the higher-scoring class, sharpening the confidence for that class; a smaller value leads to a more uniform probability distribution, increasing the uncertainty of the output of the model.

Impact of scaling factor: Changes in class probability ratios. Further analysis of the relative probability ratio between the predicted class i and other classes j is formulated as:

$$\frac{\hat{p}_i}{p_j} = \exp\left(\beta\sqrt{\frac{K}{K-1}}(\sigma_i-\sigma_j)\right). \tag{4}$$

This equation indicates that the relative relationship between the probabilities of classes is governed by the difference $\sigma_i - \sigma_j$, which is scaled by the factor $\beta\sqrt{\frac{K}{K-1}}$. In other words, the scaling factor adjusts the impact of the score difference on the final classification decision by amplifying the difference in $\sigma_i - \sigma_j$, thereby strengthening the model's ability to discriminate between classes.

It is important to note that while the scaling factor does not alter the absolute ranking of class probabilities, it significantly adjusts the relative relationships between probabilities, making the model more sensitive to the geometric structure of the classes and their distinctions.

Limit behavior analysis: from random guessing to determined. To better understand the impact of the scaling factor, consider its behavior in the limiting cases:

1. **When** $\beta\sqrt{\frac{K}{K-1}} \to 0$, the class probability distribution tends towards uniformity:

$$\lim_{\beta\sqrt{\frac{K}{K-1}}\to 0} p_{i,k} = \frac{1}{\sum_{j=1}^{K} 1} = \frac{1}{K}. \tag{5}$$

In this case, the probabilities for all classes become equal, and the model behaves as if making random guesses, unable to effectively distinguish between classes.

2. **When** $\beta\sqrt{\frac{K}{K-1}} \to \infty$, the Softmax function approaches the Argmax function, and the class probability distribution becomes nearly deterministic:

$$\lim_{\beta\sqrt{\frac{K}{K-1}}\to \infty} \hat{p}_i = \frac{1}{1 + \sum_{j\neq i}^{K} \exp(\beta\sqrt{\frac{K}{K-1}}(\sigma_j - \sigma_i))} = 1, \tag{6}$$

where i is the class with the highest score. In this extreme case, the model's output becomes nearly deterministic, always predicting the highest-scoring class.

Role of scaling factor: from adjustment to optimization. The scaling factor $\beta\sqrt{\frac{K}{K-1}}$ in the model serves more than just an adjustment function for the class probability distribution. It amplifies the score differences $\sigma_i - \sigma_j$, thereby enhancing the model's ability to distinguish between classes. Specifically, by enlarging the differences in class scores, the scaling factor increases the probability of the higher-scoring classes, improving the model's discriminative ability. When the factor is smaller, the class probabilities become more uniform, leading to weaker classification performance.

Thus, the flexible adjustment of the scaling factor allows the model to appropriately calibrate the classification confidence across different tasks and data distributions. The optimization of this factor not baseline makes the Softmax function's output more flexible but also leverages the geometric structure of the classes to enhance confidence calibration.

B. Implementation details

Experimental environment and configuration. All experiments are conducted on the Ubuntu 20.04.4 LTS operating system, Intel(R) Xeon(R) Gold 5220 CPU @ 2.20GHz with a single NVIDIA A40 48GB GPU, and 512GB of RAM. The framework is implemented with Python 3.8.19 and PyTorch 2.0.1. Other key packages include numpy 1.23.5, pandas 2.0.3, and scipy 1.10.1.

Datasets. We conduct model calibration experiments on various benchmark datasets, including CIFAR-10, CIFAR-100 [9], SVHN [14], and Tiny-ImageNet [11] aiming to evaluate calibration techniques across a spectrum of data complexities and challenges. CIFAR-10 and CIFAR-100 consist of 32×32 RGB images of 10 and 100 classes, respectively, with 50,000 training and 10,000 test samples. They are widely used as benchmarks for computer vision tasks, offering a controlled and well-understood testing environment for model calibration. SVHN comprises 32×32 images of digits (0–9) extracted from real-world street view scenes, containing 73,257 training and 26,032 test images. Its inherent variability and real-world noise introduce unique challenges for calibration studies. Tiny-ImageNet, a subset of the larger ImageNet dataset, includes 64×64 images spanning 200 classes, with 500 training, 50 validation, and 50 test samples per class. Compared to CIFAR datasets, Tiny-ImageNet demands greater generalization capabilities due to its higher resolution, broader category set, and limited training samples per class. Overall, these datasets provide a range of conditions to systematically evaluate calibration methods under varying levels of data complexity and difficulty.

To assess model robustness, we explore calibration performance under distributional shifts and out-of-distribution (OOD) scenarios. For distributional shifts, CIFAR-10 and CIFAR-100 are evaluated on their corrupted versions, CIFAR-10-C and CIFAR-100-C [6], which feature 15 corruption types (e.g., noise, blur, brightness) at 5 severity levels. For OOD detection, we test models trained on each dataset against samples from other datasets, leveraging their distinct visual and semantic characteristics. For instance, models trained on SVHN are evaluated on CIFAR-10 and CIFAR-100, while those trained on CIFAR datasets are tested on each other and SVHN. This comprehensive evaluation framework highlights the interplay between calibration quality and model generalization.

Training details. Following [8], we train a Wide Residual Network (WRN) 28-10 [24] across all datasets. Training experiments are conducted for 600 epochs using the Adam optimizer with a learning rate of 10^{-4}, applying early stopping based on validation loss. All validation experiments are derived from the training set by reserving 15% of the samples for CIFAR-10 and SVHN, and 5% for CIFAR-100. To further evaluate our method, we also train ResNet50 [5] and ResNet101 [5] backbones using the setting [12] for calibration performance. We employ the SGD optimizer with a momentum of 0.9 and an initial learning rate of 0.1. The learning rate is reduced by factor 10 at predefined intervals during training.

Reproduced details. We provide reproduced hyperparameter details of baseline methods as follows: (a) LS [18]: Following [13], we report the results obtained with $\alpha = 0.05$. (b) FL [13]: We train the models with a fixed regularization parameter $\gamma = 3$. (c) Mixup [21]: We use a hyperparameter α of 0.2, the best performing one in [21]. (d) MIT [23]: We specifically use MIT-L, with hyperparameter $\alpha = 1.0$, as applied in the main experiments of [23]. (e) FLSD [13]: Following the schedule in [13], we use the parameter $\gamma = 5$ for the samples whose output probability for the ground-truth class is within [0, 0.2), otherwise we use the parameter $\gamma = 3$. (f) CPC [1]: We set the weights of binary discrimination and binary exclusion losses as 0.1 and 1.0, respectively. (g) ACLS [16]: We set $\lambda_1 = 1.0$, $\lambda_2 = 0.1$, margin $mgn = 10.0$, and $\alpha = 0.1$ for general experiments, but adjusted mgn to 6.0 for CIFAR-10 to better suit its characteristics. (h) MMCE [10]: We report the results obtained with $\alpha = 0.5$ in CIFAR100 and $\alpha = 1.5$ in CIFAR10 and SVHN. (i) PLP [22]: We set hyperparameter γ, which determines the starting epoch at which the top layers begin to be frozen, to 1.25, as mentioned in [22]. (j) TST and VTST [8]: We set the number of neurons in the final hidden layer output to 128.

Evaluation protocols. We follow standard protocols [4, 15] and utilize the Expected Calibration Error (ECE), and Adaptive Expected Calibration Error (AECE) for evaluating network calibration performance. Expected Calibration Error (ECE) quantifies the expected $\mathbb{E}\left[|P(\hat{y}_i = y_i|\hat{p}_i) - \hat{p}_i|\right]$ of the absolute difference between predicted confidence and actual accuracy. ECE is calculated by dividing the confidence range $(0, 1]$ into 15 equally spaced bins and averaging the absolute differences within each bin, weighted by bin size:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \Big| Acc(B_m) - Conf(B_m) \Big|, \tag{7}$$

where $|B_m|$ is the number of samples in bin m, n is the total number of samples, $Acc(B_m)$ denotes the accuracy within bin B_m, and $Conf(B_m)$ represents the average confidence within the bin. Bins are uniformly spaced over the range $(0, 1]$, ensuring a consistent confidence interval width of $1/M$. Following standard practice, we set bins $M = 15$. The difference between Acc and $Conf$ can indicate the calibration gap for model calibration. Adaptive Expected Calibration Error (AECE) enhances ECE by dynamically adjusting the bin boundaries through the confidence distribution of the samples. In contrast to ECE with fixed bins, AECE employs an adaptive binning strategy where bin sizes are determined to evenly distribute samples across bins. This approach facilitates a more effective assessment of calibration performance, particularly for imbalanced or skewed confidence distributions. Besides, Top-1 classification accuracy is reported for discriminative evaluation. For OOD detection, we also compare AUROC and FPR95, adhering to standard evaluation protocols [17].

	CIFAR-10				**CIFAR-100**				**SVHN**			
	baseline		+Ours		baseline		+BalCAL		baseline		+BalCAL	
	Acc↑	ECE↓	Acc↑	ECE↓	Acc↑	ECE↓	Acc↑	ECE↓	Acc↑	ECE↓	Acc↑	ECE↓
ACLS[16]	93.12	5.26	93.07↓	2.10↓	72.36	13.08	72.36↑	2.07↓	95.33	3.20	95.35↑	1.47↓
LS[18]	92.12	2.79	92.16↑	2.88↑	73.58	6.03	73.60↑	3.86↓	95.50	3.77	95.52↓	1.52↓
FL[13]	92.11	1.99	92.06↓	1.32↓	71.54	14.26	72.06↑	1.16↓	94.79	1.24	94.71↓	1.70↑
FLSD[13]	92.42	1.72	92.35↓	1.16↓	71.19	14.20	71.59↑	1.08↓	94.60	1.03	94.68↑	1.74↓
CPC[1]	91.56	6.28	91.71↑	2.47↓	73.33	12.53	73.31↓	7.95↓	94.73	1.61	94.83↑	0.85↓
MMCE[10]	90.04	3.60	90.24↑	1.72↓	68.77	17.18	70.28↑	3.09↓	93.99	2.30	94.60↑	2.76↑
Mixup[21]	94.71	2.56	94.72↑	1.25↓	77.05	4.51	76.40↓	2.27↓	94.82	3.75	92.38↓	1.46↓
MIT[23]	94.27	2.18	94.72↑	0.69↓	73.38	12.31	75.05↑	3.06↓	93.69	0.78	94.22↑	0.75↓

Table 1. Results of combination ours with commonly baseline calibration methods. Green arrows indicate improved performance, while red arrows indicate decreased performance.

C. More evaluation results

Combination ours with baseline calibration methods. We combine BalCAL with common baseline calibration methods, which consist of loss-based(ACLS[16], LS[18], FL[13], FLSD[13], CPC[1], MMCE[10]) and augment-based (Mixup[21] and MIT[23]) methods in Table 1. We can find that model performance presents improvements on most baseline methods, especially ECE. This indicates that our method has strong compatibility and the potential to enhance model performance.

Extended Architectures and Datasets We conducted extensive experiments on additional model architectures, including WRN-26-10 [24], DenseNet-121 [7], and ViT-B_16 [3], as well as on larger datasets such as ImageNet [2]. We also incorporated new comparative methods [19, 20] to further validate our approach. The results are presented in Tab. 2 and Tab. 3, where the experimental setup for Tab. 2 follows [13], and the setup for Tab. 3 adheres to [3]. Our method consistently demonstrates superior performance across all evaluated scenarios.

Method	WRN-26-10				DenseNet-121			
	CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100	
	Acc↑	ECE↓	Acc↑	ECE↓	Acc↑	ECE↓	Acc↑	ECE↓
Vanilla	96.03	3.37	79.82	10.98	95.05	4.02	77.73	12.07
DualFocal*	96.04	0.81	80.09	1.79	94.57	0.57	77.60	1.81
PCS*	-	0.99	-	1.92	-	0.78	-	2.75
Ours	**96.24**	**0.79**	**80.27**	**1.51**	**95.35**	**0.53**	**77.81**	**1.77**

Table 2. Calibration Performance Across Diverse Model Architectures. * denotes paper results.

Methods	CIFAR-100			ImageNet		
	Acc↑	ECE↓	**AECE↓**	Acc↑	ECE↓	**AECE↓**
Vanilla	92.67	1.75	1.45	81.85	5.74	5.72
MMCE[10]	**92.76**	1.27	1.08	81.68	4.91	4.86
DualFocal	92.73	1.25	1.05	81.94	1.69	1.61
TST[8]	92.10	1.09	1.27	81.04	4.38	4.31
VTST[8]	92.67	1.49	1.34	80.00	2.01	1.91
Ours	92.69	**0.96**	**0.97**	**82.32**	**1.48**	**1.49**

Table 3. Calibration Analysis on ViT-B_16.

Uncertainty evaluation in OOD detection. In Table 4, we examine model confidence and entropy for OOD samples. Ideally, OOD samples should present low confidence and high entropy, reflecting the model's caution with unknown data. Our method can align with this trend and achieve the best performance with lower confidence and higher entropy, indicating significant uncertainty regarding unfamiliar samples. Such uncertainty enhances the model's self-correction ability, improving adaptability in OOD detection and ensuring reliable performance across diverse samples.

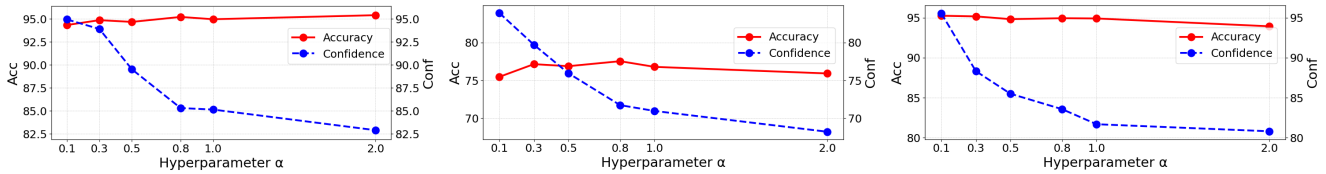
Methods	CIFAR-10				CIFAR-100				SVHN			
	SVHN		CIFAR-100		SVHN		CIFAR-10		CIFAR-10		CIFAR-100	
	Conf↓	Entropy↑	Conf↓	Entropy↑	Conf↓	Entropy↑	Conf↓	Entropy↑	Conf↓	Entropy↑	Conf↓	Entropy↑
Vanilla	86.35	34.96	89.69	27.02	86.17	40.02	83.24	47.72	81.65	51.08	81.38	51.41
ACLS[16]	90.55	26.74	90.73	26.56	68.33	134.92	67.73	136.49	77.95	66.44	79.18	62.84
LS[18]	85.76	53.63	85.33	55.88	52.89	238.43	55.55	226.29	65.93	116.51	67.47	112.34
mixup[21]	82.04	56.27	81.51	61.30	64.21	149.04	59.25	175.28	60.38	121.14	62.16	116.57
MIT[23]	71.20	77.50	79.03	56.81	59.67	133.32	67.18	106.62	79.03	56.37	77.40	60.68
FL[13]	75.52	63.65	75.07	65.44	69.43	96.71	70.82	89.72	62.97	106.21	63.79	104.49
FLSD[13]	73.15	69.53	75.42	64.58	69.10	101.06	70.10	91.70	66.39	95.83	66.45	95.61
CPC[1]	89.64	37.11	86.97	45.70	66.52	171.02	60.79	200.46	57.38	140.76	57.67	140.22
PLP[22]	87.55	32.94	86.85	35.59	70.25	110.65	68.29	113.99	81.84	50.55	82.32	49.34
MMCE[10]	66.51	97.36	71.26	84.60	68.71	102.19	71.71	91.51	36.41	185.99	36.84	184.77
TST[8]	68.51	95.73	72.08	84.35	60.95	144.44	58.13	149.60	61.80	118.68	61.93	118.30
VTST[8]	76.34	71.87	77.12	70.79	54.50	191.72	54.78	184.83	70.71	87.10	70.99	86.85
Ours	**64.74**	**120.08**	**68.79**	**105.86**	**36.66**	**305.63**	**45.07**	**257.74**	**60.27**	**130.03**	**60.97**	**127.84**

Table 4. Confidence and entropy analysis for OOD samples.

Underconfidence problem with mixup Mixup is a data augmentation method that improves accuracy by taking convex combinations of pairs of examples and labels. Given a sample (x_i, y_i), Mixup generates a new sample by mixing it and another sample (x_j, y_j) as follows:

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \quad \tilde{y} = \lambda y_i + (1-\lambda)y_j, \tag{8}$$

where (x_i, y_i) and (x_j, y_j) are randomly sampled from the training set D, and $\lambda \in [0, 1]$ is a mixing coefficient sampled from the Beta distribution Beta(α, α). This process applies the same mixing coefficient λ to both the input samples and their corresponding labels. The hyperparameter α controls the interpolation strength between the input pairs and their labels.



(a) CIFAR-10 (b) CIFAR-100 (c) SVHN

Figure 1. Effect of α on model confidence and underconfidence.

Recent work (MIT) [23] has indicated that Mixup improves model calibration performance but can lead to underconfidence problems. As illustrated in Fig. 1, we can find that when α becomes large, model confidence rapidly decreases and exhibits the underconfidence phenomenon. To investigate whether our approach effectively mitigates the underconfidence problem with Mixup, we evaluate the accuracy and ECE results of Mixup, MIT, and Mixup+Ours with various settings in Tab. 5. Mixup+Ours method significantly surpasses Mixup and MIT under different ratios α across various datasets.

Dataset	α	Mixup		MIT		**Mixup+Ours**	
		ACC↑	ECE↓	ACC↑	ECE↓	ACC↑	ECE↓
CIFAR-10	0.1	94.30	2.39	93.67	3.62	**94.72**	**1.25**
	0.3	94.83	3.38	94.40	2.62	**95.28**	**1.13**
	0.5	94.64	6.00	93.29	2.97	**94.78**	**1.39**
	0.8	**95.18**	10.22	93.60	2.68	95.05	**1.71**
	1.0	94.93	9.93	94.27	2.18	**95.06**	**1.21**
	2.0	**95.37**	12.50	93.10	2.61	95.13	**1.37**
CIFAR-100	0.1	75.48	8.48	73.48	14.18	**76.40**	**2.27**
	0.3	77.16	2.64	74.23	12.41	**78.22**	**1.89**
	0.5	76.89	3.97	74.55	11.11	**78.15**	**2.47**
	0.8	77.55	6.12	73.25	12.00	**77.88**	**2.93**
	1.0	76.80	6.43	73.38	12.31	**78.06**	**2.16**
	2.0	75.93	8.64	72.16	11.44	**76.68**	**1.67**
SVHN	0.1	95.25	1.65	95.02	1.76	**95.47**	**1.46**
	0.3	95.17	6.92	94.21	**0.55**	**95.37**	1.13
	0.5	94.82	9.36	94.10	1.14	**95.39**	**1.13**
	0.8	94.94	11.46	93.91	**0.83**	**95.80**	1.01
	1.0	94.91	13.22	93.69	**0.78**	**95.84**	1.30
	2.0	93.93	13.13	93.11	**1.43**	**95.65**	3.16

Table 5. Comparison Performance across datasets under different ratios α.

Computational cost We rigorously evaluate the training efficiency by comparing the per-epoch time costs between our method and vanilla implementations across multiple architectures and datasets. As systematically documented in Tab. 6, our approach improves calibration performance at an acceptable cost.

	WRN-26-10		DenseNet-121	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
Vanilla	63.06	63.84	54.01	54.43
Ours	65.73	67.64	55.91	57.79

Table 6. Computational cost (s) per training epoch.

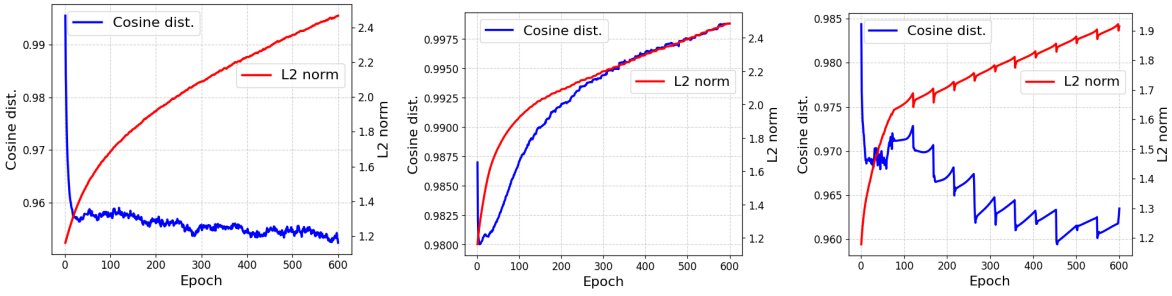
Dynamic and Fixed γ: In this paper, we employ a dynamic γ, derived from Eq. (7). The parameter γ in Eq. (6) and Eq. (8) balances the learnable classifier and the fixed ETF classifier. Our ablation study in Tab. 7 indicates that γ in L_{total} is relatively insensitive, and a fixed γ (e.g., 0.5) achieves competitive performance. However, using a dynamic γ in L_{total} provides slight improvements without additional computational cost. In contrast, for $\boldsymbol{p}^{fused}$, γ is more sensitive and dataset-dependent, making a dynamic γ essential for performance enhancement. To ensure consistency, we adopt the same dynamic γ for both L_{total} and $\boldsymbol{p}^{fused}$, although its impact is more pronounced in $\boldsymbol{p}^{fused}$.

γ of	γ of	CIFAR-10		CIFAR-100		**SVHN**	
L_{total}	$\boldsymbol{p}^{fused}$	ACC↑	ECE↓	ACC↑	ECE↓	ACC↑	ECE↓
fixed	fixed	91.95	1.14	73.12	4.94	95.40	0.44
fixed	dynamic	92.12	0.83	72.74	**3.97**	95.42	0.39
dynamic	dynamic	**92.23**	**0.76**	**73.21**	4.21	**95.47**	**0.24**

Table 7. Ablation study on the impact of γ.

Evolution of prototype distances We analyze the evolution of prototype representations between two classifiers during training, as illustrated in Figure 2. For CIFAR-10 and SVHN datasets, the cosine distance between prototypes steadily decreased, indicating improved alignment of class representations, while the L_2 distance consistently increased, reflecting an expansion in prototype distributions. Conversely, the CIFAR-100 dataset exhibited a distinct trend: the cosine distance sharply declined in the early stages, followed by a gradual rise, likely due to the dataset's complex and dispersed class structures. These observations highlight the significant impact of dataset-specific feature distributions on learning dynamics. For CIFAR-10 and SVHN, reduced cosine distance facilitated consistent high-level feature representations, enhancing class separability. Simultaneously, increased L_2 distance broadened prototype distributions, contributing to varied confidence outputs between classifiers. On CIFAR-100, the rapid initial alignment in cosine distance was followed by divergence, as seen in the increasing L_2 distance, signaling a more dispersed feature space.

By leveraging these dynamics, the growing L_2 distance, indicating the difference in output confidence between the two classifiers, serves as a foundation to balance learnable and ETF classifiers, refining calibration. Using a shared feature extractor and distinct learning strategies, the complementary interactions between classifiers improve model calibration, robustness, and accuracy.



(a) CIFAR-10 (b) CIFAR-100 (c) SVHN

Figure 2. Dynamic changes of cosine and L2 distances between classifier prototypes during training.

Visualization results and analysis We present t-SNE visualizations of more training-time methods on CIFAR-10, as shown in Figure 3. We observe that our method yields well-clustered intra-class representations and more discriminative inter-class distances. This demonstrates that our approach effectively enhances representation learning and improves the balance between confidence and accuracy, which is crucial for addressing the overconfidence and underconfidence issues commonly encountered in model calibration.

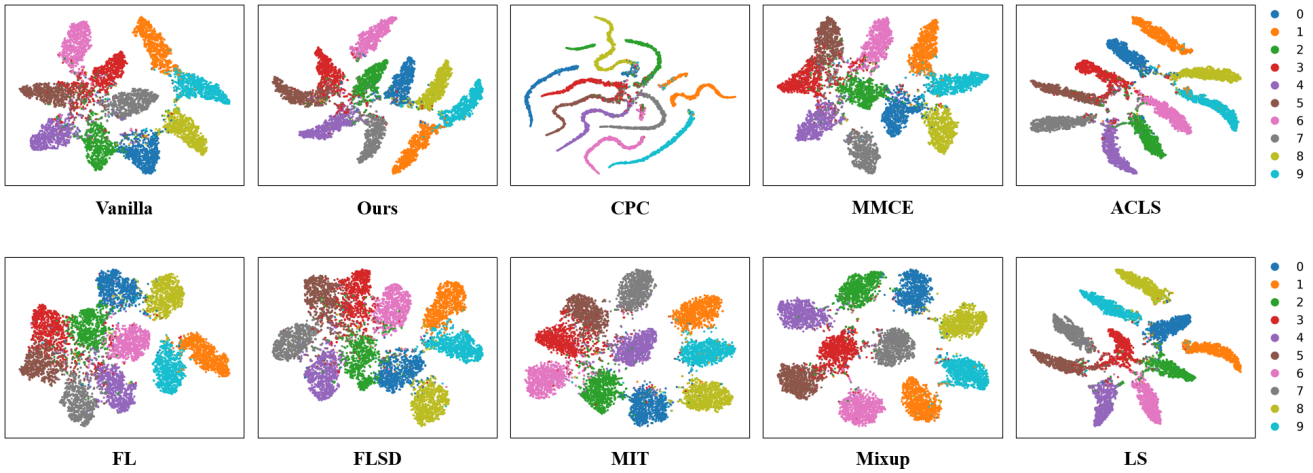


Figure 3. t-SNE visualizations of representations on CIFAR-10.

References

[1] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2019. 3, 4

[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] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 4

[4] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017. 3

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[6] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 2

[7] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4

[8] Mikkel Jordahn and Pablo M. Olmos. Decoupling feature extraction and classification layers for calibrated neural networks. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. 3, 4

[9] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 2

[10] Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018. 3, 4

[11] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 2

[12] Bingyuan Liu, Ismail Ben Ayed, Adrian Galdran, and Jose Dolz. The devil is in the margin: Margin-based label smoothing for network calibration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[13] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating deep neural networks using focal loss. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 3, 4

[14] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *Proceedings of the 2011 International Conference on Machine Learning (ICML)*, 2011. 2

[15] Khanh Nguyen and Brendan O'Connor. Posterior calibration and exploratory analysis for natural language processing models. *arXiv preprint arXiv:1508.05154*, 2015. 3

[16] H. Park, J. Noh, Y. Oh, D. Baek, and B. Ham. Acls: Adaptive and conditional label smoothing for network calibration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 3, 4

[17] Francesco Pinto, Harry Yang, Ser-Nam Lim, Philip Torr, and Puneet K Dokania. Using mixup as a regularizer can surprisingly improve accuracy & out-of-distribution robustness. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 3

[18] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3, 4

[19] Linwei Tao, Minjing Dong, Daochang Liu, Changming Sun, and Chang Xu. Calibrating a deep neural network with its predecessors. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2023. 4

[20] Linwei Tao, Minjing Dong, and Chang Xu. Dual focal loss for calibration. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 33833–33849, 2023. 4

[21] Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 3, 4

[22] Deng-Bao Wang and Min-Ling Zhang. Calibration bottleneck: Over-compressed representations are less calibratable. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. 3, 4

[23] Deng-Bao Wang, Lanqing Li, Peilin Zhao, Pheng-Ann Heng, and Min-Ling Zhang. On the pitfall of mixup for uncertainty calibration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7609–7618, 2023. 3, 4, 5

[24] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, page 87, York, UK, 2016. BMVA Press. 3, 4