Consistency Regularization for Training Confidence-Calibrated Classifiers

Youngbum Hur [*1] **Jihoon Tack** [*2] **Eunho Yang** [2 3] **Sung Ju Hwang** [2 3] **Jinwoo Shin** [2]

Abstract

Out-of-distribution (OOD) detection, *i.e.*, identifying whether a given test sample is drawn from outside the training distribution, is essential for a deep classifier to be deployed in a real-world application. The existing state-of-the-art methods of OOD detection tackle this issue by utilizing the internal feature of the classification network. However, we found that such detection methods inherently struggle to detect hard OOD images, *i.e.*, drawn near from the training distribution: a naive softmax-based baseline even outperforms them. Motivated by this, we propose a simple yet effective training scheme for further calibrating the softmax probability of a classifier to achieve high OOD detection performance under both hard and easy scenarios. In particular, we suggest to optimize a consistency regularization loss during training, which injects a strong inductive bias by forcing the network prediction to be consistent over data augmentations. Our experiments demonstrate the superiority of our simple method under various OOD detection scenarios.

1. Introduction

Deep neural networks (DNNs) have demonstrated remarkable performance on many classification tasks such as image classification (Girshick, 2015), medical diagnosis (Caruana et al., 2015), and video prediction (Villegas et al., 2017). However, it is widely known that well trained deep classifiers are often overconfident even for novel examples, unseen during training (Guo et al., 2017). This can become a serious problem when deployed in real-world systems (Yampolskiy and Spellchecker, 2016).

Given a DNN classifier, the conventional way for out-of-distribution (OOD) detection is to detect sample with a low prediction confidence, *i.e.*, the maximum softmax probability among classes (Hendrycks and Gimpel, 2017; Liang et al., 2018). Recently, more advanced OOD detection methods (Lee et al., 2018; Sastry and Oore, 2020) design a detection score using the information of internal layers and claim that can achieve near-optimal detection rates.

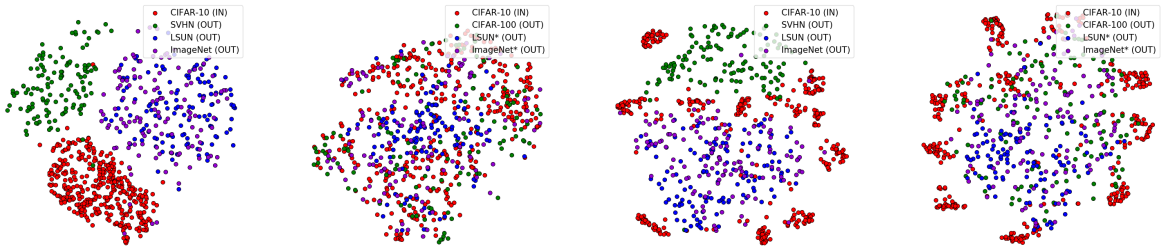
Contribution. We first found that the existing state-of-the-art methods utilizing internal layers perform poorly on *hard OOD datasets*, *i.e.*, drawn near from the in-distribution: they perform even worse than the conventional method that uses the softmax probability. This is because such hard OOD samples are not clearly separable in the internal layers as they use similar (low-level) features with the in-distribution samples. Instead, the softmax probability is rather more informative to detect them, as it captures the most discriminative structural (high-level) information. This motivates us to revisit the question, how to calibrate the softmax probability of a deep classifier for improved OOD detection.

In this paper, we suggest to optimize a simple auxiliary *consistency regularization* term during training for a better calibrated softmax probability for OOD detection. To be specific, the regularization scheme forces the network to predict the consistent softmax probability over data augmentations. Intuitively, such regularization injects a strong inductive bias to the model itself, hence, learns a more informative softmax probability. We show that such inductive bias lead to a improved calibration for OOD detection. To further utilize the effect of our training scheme, we also consider a test-time augmentation scheme for the inference. Finally, we show that our method can be further enhanced using a broader class of augmentations, *e.g.*, rotation, whose predictions are not necessarily consistent.

We verify the efficacy of our method under various environments of detecting OOD on CIFAR-10/100 (Krizhevsky et al., 2009) and CUB-200 (Wah et al., 2011). Overall, our method achieves high performance for all tested datasets and especially shows robust results in hard OOD detection scenarios. In particular, our method improves the AUROC, compared to the baseline: 86.36% to 92.97% on CIFAR-10, when CIFAR-100 is considered as OOD. We also demonstrate that our method improves the expected calibration error (ECE) (Guo et al., 2017), compared to the cross-entropy training by 20.35% to 6.78% on CUB-200.

[*]Equal contribution [1]Samsung Advanced Institute of Technology, Suwon, South Korea [2]Korea Advanced Institute of Science and Technology, Daejeon, South Korea [3]AITRICS, Seoul, South Korea. Correspondence to: Jinwoo Shin <jinwoos@kaist.ac.kr>.

Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning., Copyright 2021 by the author(s).



(a) Internal: easy OOD (b) Internal: hard OOD (c) Penultimate: easy OOD (d) Penultimate: hard OOD

Figure 1. The t-SNE visualization of the internal (residual block 2) and penultimate feature of ResNet-34 pre-trained on CIFAR-10. SVHN, resized LSUN and ImageNet are used as easy OODs, and CIFAR-100, fixed version of LSUN and ImageNet are used as hard OODs. * denotes the fixed version dataset.

Table 1. AUROC (%) of ResNet-18 trained on one-class remove CIFAR-10: samples from the given class are considered as OOD, while the remaining samples of the nine classes are considered for training (in-distribution). "Inter." denotes the methods that utilize the internal feature of the network. "Ours" indicates the network trained on our objective (5), and tested with the proposed inference method (6). The final column indicates the mean AUROC across all the classes and the bold denotes results within 1% from the highest result.

Method	Inter.	Plane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck	Mean
Baseline (Hendrycks and Gimpel, 2017)	-	86.53	**74.67**	87.70	85.76	89.01	77.60	88.30	85.92	78.18	**82.86**	83.65
Mahalanobis (Lee et al., 2018)	✓	72.00	58.13	65.19	61.24	51.99	57.57	54.50	57.78	49.54	43.60	57.15
Gram matrix (Sastry and Oore, 2020)	✓	78.28	62.63	73.04	65.77	72.31	57.68	69.27	62.39	63.06	55.65	66.01
Ours	-	**91.57**	**75.10**	**94.09**	**90.68**	**93.94**	**85.16**	**94.15**	**87.67**	**89.40**	**82.24**	**88.40**

2. Towards better OOD benchmarks

In this section, we point out that existing state-of-the-art methods (Lee et al., 2018; Sastry and Oore, 2020) in out-of-distribution (OOD) detection do not generalize well on detecting hard OOD datasets and show that the cause lies in their usages of the internal layer of the network.

2.1. Easy and hard OOD datasets

At a high level, we denote easy OOD as a sufficiently far away distribution from the training distribution, while hard OOD is not[1].We remark that detecting hard OOD is an important and essential component for a real-world deployment. To this end, we propose a new hard OOD detection scenario coined "*one-class remove CIFAR-10*" which has never explored in the literature. In this setup, a given class labeled samples of CIFAR-10 (Krizhevsky et al., 2009) becomes the OOD dataset, and samples with the remaining nine classes are used as training or in-distribution.

As shown in the Table 1, one can observe that the state-of-the-art methods (Lee et al., 2018; Sastry and Oore, 2020) show poor performance in all cases, while the simple baseline with maximum softmax probability (Hendrycks and Gimpel, 2017) significantly and consistently outperforms the others. This motivates us take a closer look at the issue: the recent methods commonly utilize the ensembles of the internal layer feature for OOD detection.

[1]Tack et al. (2020) demonstrated that such easy OOD can be detected with simple statistics, *i.e.*, without deep networks.

2.2. Analysis of the internal representation

To analyze the internal layer of the network, we visualize the data representation with t-SNE. We select CIFAR-10 as in-distribution and train ResNet-34 (He et al., 2016) with standard cross-entropy loss. Then we choose SVHN (Netzer et al., 2011), resized LSUN, and ImageNet (Liang et al., 2018) as easy OOD where such datasets are known to be easily detected with simple statistics (Tack et al., 2020). For hard OOD, we use CIFAR-100 (Krizhevsky et al., 2009), fixed versions of LSUN, and ImageNet (Tack et al., 2020).

One can observe that easy OOD datasets are clearly separated with in-distribution datasets while hard OOD are still entangled in the internal layer (see Figure 1a, 1b). This observation aligns with our claim that the internal layer cannot capture the difference between hard OOD and in-distribution, as they are naturally expected to have similar low-level features. Meanwhile, both easy and hard OOD are separated well from the in-distribution in the penultimate layer of the network (see Figure 1c, 1d), as it captures the most discriminative structural (high-level) information. We also support the claim with quantitative measurement of in-and-out distribution separation (see Appendix C).

3. Consistency regularization for training confidence-calibrated classifiers

We consider a classification task with K classes dataset $\mathcal{D} = \{(x_m, y_m)\}_{m=1}^{M} \subseteq \mathcal{X} \times \mathcal{Y}$ where $x \in \mathbb{R}^d$ represents an input sampled from a certain data-generating distribution P in an *i.i.d.* manner, and $\mathcal{Y} := \{1, \dots, K\}$ represents a set

of possible class labels. Let $p_\theta : \mathbb{R}^d \to \Delta^{K-1}$ be a neural network modeled to output a probability simplex $\Delta^{K-1} \in \mathbb{R}^K$, *e.g.*, via a softmax layer. The goal of our work is to train a classifier with better calibrated softmax probability for out-of-distribution (OOD) detection *i.e.*, for a given OOD sample $x_{\texttt{out}}$, the classifier shows high entropy than the in-distribution sample $x_{\texttt{in}} \sim P$, $\mathcal{H}(p_\theta(x_{\texttt{out}})) > \mathcal{H}(p_\theta(x_{\texttt{in}}))$ where $\mathcal{H}$ is the entropy of the probability; $\mathcal{H}(p_\theta(x)) := -\sum_{i=1}^{K} p^{(i)}(x) \log(p^{(i)}(x))$ where $p^{(i)}(x)$ is the prediction probability for class i.

3.1. Training: consistency regularization

We suggest to optimize a simple auxiliary *consistency regularization* during training to improve the calibration of a deep classifier for a better OOD detection. Specifically, the proposed term forces the predictive distributions of data augmentations to be consistent. While such regularization is commonly used in other different domains, *e.g.*, training GANs (Zhang et al., 2020) or semi-supervised learning (Xie et al., 2020), it is still unknown whether such regularization will be is useful for OOD detection, which we essentially investigate in this paper. Our intuition is that the model learns more informative softmax probability, hence, leads to improved calibration for OOD detection.

We simply adapt one of the most commonly used consistency regularization scheme (Xie et al., 2020) which utilizes a weak and strong augmentation set, *i.e.*, $\mathcal{T}_{\texttt{weak}}$ and $\mathcal{T}_{\texttt{strong}}$, where strong augmentation are likely to shift the input distribution. For a given labeled input $(x, y) \sim \mathcal{D}$, and randomly sampled augmentation $T_{\texttt{w}} \sim \mathcal{T}_{\texttt{weak}}, T_{\texttt{s}} \sim \mathcal{T}_{\texttt{strong}}$, the regularization loss is as follows:

$$\mathcal{L}_{\texttt{con}}(x) := \texttt{KL}\Big(p_{\tilde{\theta}}\big(T_{\texttt{w}}(x)\big) \parallel p_\theta\big(T_{\texttt{s}}(x)\big)\Big), \tag{1}$$

where KL denotes the Kullback-Leibler (KL) divergence, and $\tilde{\theta}$ is a fixed copy of parameter θ. Note that we do not consider advanced techniques from (Xie et al., 2020), *e.g.*, loss scheduling, for the simplicity.

3.2. Inference: Augment-Entropy

We find one can further the softmax probability calibration of the network by ensembling it over random augmentation $T(x)$ where $T \sim \mathcal{T}$; we denote such technique as *Augment-Entropy*. Concretely, for a given test input x and a pre-trained classifier θ, we calculate the following score:

$$s_{\texttt{aug-ent}}(x) := \mathcal{H}\Big(\mathbb{E}_{T\sim\mathcal{T}}\Big[p_\theta\big(T(x)\big)\Big]\Big), \tag{2}$$

where $\mathcal{T}$ is a pre-defined augmentation family, and $\mathcal{H}$ is the entropy function. We approximate the proposed score (2) via Monte Carlo integration with n randomly sampled augmentations from $\mathcal{T}$. See the details of augmentation policy and sampling number in Appendix A.

3.3. Extension: distribution augment

One can further enhance the model calibration by adapting broader class of data augmentations, *e.g.*, rotation, whose predictions are not necessarily consistent. We denote such augmentation as distribution augment (Jun et al., 2020) which consist of S different transformation including identity I, *i.e.*, $\mathcal{T}_{\texttt{dist}} := \{T_{\texttt{d}}^0 = I, T_{\texttt{d}}^1, \dots, T_{\texttt{d}}^{S-1}\}$. We suggest to use distribution augment in two aspects, (a) for self-supervised learning (Hendrycks et al., 2019) and (b) as an unlabeled training sample. While self-supervised learning is previously investigated as an effective approach for improving OOD detection, we believe rethinking distribution augment as an unlabeled sample is an interesting direction: it is known to be harmful when the distribution augmented sample is forced as the original label (Lee et al., 2020).

For a given input $(x, y) \sim \mathcal{D}$, and a distribution augment $\mathcal{T}_{\texttt{dist}}$, the goal of self-supervised loss is to classify the applied augmentation $T_{\texttt{d}} \in \mathcal{T}_{\texttt{dist}}$ of the given sample. For the auxiliary loss of unlabeled distribution augment (*i.e.*, the unlabeled loss), we utilize $\mathcal{L}_{\texttt{con}}$ (1). Namely, for a distribution augment prediction classifier $p_{\texttt{d}}$: which shares the same penultimate feature with p_θ, and standard cross-entropy $\mathcal{L}_{\texttt{CE}}$, the self-supervised loss and unlabeled loss as follow:

$$\mathcal{L}_{\texttt{self}}(x) := \frac{1}{S} \sum_{T_{\texttt{d}} \in \mathcal{T}_{\texttt{dist}}} \mathcal{L}_{\texttt{CE}}\Big(p_{\texttt{d}}\big(T_{\texttt{d}}(x)\big), T_{\texttt{d}}\Big), \tag{3}$$

$$\mathcal{L}_{\texttt{unlabel}}(x) := \frac{1}{S-1} \sum_{T_{\texttt{d}} \in \mathcal{T}_{\texttt{dist}} \setminus \{I\}} \mathcal{L}_{\texttt{con}}\big(T_{\texttt{d}}(x)\big). \tag{4}$$

Overall objective. Finally, the final objective of the extended version of our proposed method can be defined by simply combining the defined objectives (1), (3), (4) with the standard cross-entropy loss:

$$\begin{aligned}\mathcal{L}_{\texttt{final}}(x, y) :=& \mathcal{L}_{\texttt{CE}}(x, y) + \mathcal{L}_{\texttt{con}}(x) \\ &+ \mathcal{L}_{\texttt{self}}(x) + \mathcal{L}_{\texttt{unlabel}}(x)\end{aligned} \tag{5}$$

Overall inference score. We simply adapt the self supervised loss (3) as an additional detection score along with the Augment-Entropy (2). Namely, the new inference score is:

$$s_{\texttt{final}}(x) := s_{\texttt{aug-ent}}(x) + \mathcal{L}_{\texttt{self}}(x). \tag{6}$$

4. Experiments

We verify the efficacy of our method under various environments of detecting out-of-distribution (OOD) on CUB-200 (Wah et al., 2011) (see Appendix D for CIFAR-10/100 (Krizhevsky et al., 2009) results). For the evaluation, we mainly report the area under the receiver operating characteristic curve (AUROC) as a threshold-free evaluation metric for a detection score, and the expected calibration error

Table 2. AUROC (%) of ResNet-18 trained on CUB-200. Here, we consider various *hard OOD* detection scenarios. "CE" denote the cross-entropy. The final column indicates the mean AUROC across all the OOD datasets and the bold denotes the best results.

		CUB-200 →					
Train	Inference	Food-101	Caltech-256	MIT-67	Places-365	Dogs	Mean
CE	Baseline (Hendrycks and Gimpel, 2017)	72.71	73.41	75.16	75.03	75.09	74.28
CE	ODIN (Liang et al., 2018)	79.80	79.95	82.47	81.99	79.58	80.76
CE	Mahalanobis (Lee et al., 2018)	84.20	81.24	78.70	76.36	62.32	76.56
CE	Gram matrix (Sastry and Oore, 2020)	81.04	81.74	74.64	74.00	72.15	76.71
CSI (Tack et al., 2020)	Baseline (Hendrycks and Gimpel, 2017)	75.99	76.81	84.45	81.22	69.09	77.51
CSI (Tack et al., 2020)	CSI-ens (Tack et al., 2020)	82.33	82.84	92.97	89.64	76.34	84.82
Ours, $\mathcal{L}_{\texttt{final}}$ (5)	Ours, $s_{\texttt{final}}$ (6)	**97.12**	**96.45**	**97.98**	**97.01**	**97.38**	**97.19**

Table 3. ECE (%) of ResNet-18/34 trained on CIFAR-10 and CUB-200, respectively. Parentheses indicate the relative rate of ECE from the cross-entropy (CE), and bold denotes the best results.

Train method\ dataset	CIFAR-10	CUB-200
CE	3.64	20.35
CE + Self-sup (3)	2.65	17.49
CSI (Tack et al., 2020)	2.37	26.21
Ours, $\mathcal{L}_{\texttt{final}}$ (5)	**1.51 (-58.51%)**	**6.78 (-66.68%)**

(ECE) (Naeini et al., 2015). Here, ECE estimates whether a classifier can indicate when they are likely to be incorrect for test samples (from in-distribution) by measuring the difference between prediction confidence and accuracy. The formal description of the metrics can be found in Appendix A. Overall, our results clearly demonstrate that consistency regularization for classifiers improves the performance of detecting hard OOD samples and shows significant improvement of the confidence calibration in all tested datasets. We also perform an ablation study in Appendix E.

We consider inference baselines, including, baseline (Hendrycks and Gimpel, 2017), ODIN (Liang et al., 2018), Mahalnobis (Lee et al., 2018), and Gram-matrix (Sastry and Oore, 2020). For training baselines, we consider self-supervised learning (Hendrycks et al., 2019), and CSI (Tack et al., 2020). See Appendix B for detailed descriptions.

For augmentation policy, we use Inception crop (Szegedy et al., 2015), horizontal flip for the weak augmentation $\mathcal{T}_{\texttt{weak}}$ and RandAugment (Cubuk et al., 2019) for the strong augmentation $\mathcal{T}_{\texttt{strong}}$ by following (Xie et al., 2020). For distribution augment $\mathcal{T}_{\texttt{dist}}$, we use random rotation of $0°, 90°, 180°, 270°$, by following (Hendrycks et al., 2019). See Appendix A for details of training and evaluation.

4.1. Main results

Fine-grained classification. We consider CUB-200 (Wah et al., 2011) as in-distribution while the following datasets are used as out-of-distribution: Food-101 (Bossard et al., 2014), Caltech-256 (Griffin et al., 2007), MIT-67 (Quattoni and Torralba, 2009), and Dogs (Khosla et al., 2011). Overall, our proposed method significantly outperforms prior methods in all datasets tested as shown in Table 2. We remark that consistency regularization significantly and consistently improves the performance of OOD detection. Interestingly, one can observe that the state-of-the-art method CSI is not effective in this scenario. We conjecture that the contrastive learning performance is sensitive to the data augmentation policy and should use the suitable policy for each dataset: the proposed policy from CSI fails to generalize in the current setup. On the other hand, our method performs well, even in the fine-grained datasets. We also observed that our method shows comparable performance with CSI on CIFAR datasets while it has significantly low training cost campare to CSI (see Appendix D).

Calibration. We demonstrate the effectiveness of our method on the calibration performance under various image classification datasets: CIFAR-10, and CUB-200 (see Appendix D for CIFAR-100 results). As shown in Table 3, the proposed consistency regularization significantly and consistently improves the confidence calibration. We note that the proposed method is specialized for calibration compared to other baselines. Moreover, we observed that our method also increase the classification accuracy which is somewhat interesting (see Appendix D). In the ablation study, we find that consistency regularization loss (1) and self-supervised loss (3) is effective for improving both classification accuracy and calibration, while unlabeled loss (4) is specialized for improving the calibration (see Appendix E).

5. Conclusion

In this paper, we show that the existing OOD detection methods that are using the internal feature of the network perform poorly for hard OOD images. Motivated by this, we propose a simple yet effective training scheme for better OOD detection and network calibration. We evaluate our methods under various scenarios and show that our proposed method is robust for easy and hard OOD datasets. Due to the simplicity of our method, we think it could enjoy a broader usage under various applications in the future.

References

David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. In *International Conference on Learning Representations*, 2020.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *ACM SIGKDD*, 2015.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019.

Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout, 2017.

Ross Girshick. Fast r-cnn. In *IEEE International Conference on Computer Vision*, 2015.

Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset, 2007.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017.

Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In *Advances in Neural Information Processing Systems*, 2019.

Heewoo Jun, Rewon Child, Mark Chen, John Schulman, Aditya Ramesh, Alec Radford, and Ilya Sutskever. Distribution augmentation for generative modeling. In *International Conference on Machine Learning*, 2020.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

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

Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. Self-supervised label augmentation via input transformations. In *International Conference on Machine Learning*, 2020.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, 2018.

Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI Conference on Artificial Intelligence*, 2015.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems*, 2011.

Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420. IEEE, 2009.

Chandramouli Shama Sastry and Sageev Oore. Detecting out-of-distribution examples with gram matrices. In *International Conference on Machine Learning*, 2020.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. Csi: Novelty detection via contrastive learning on distributionally shifted instances. In *Advances in Neural Information Processing Systems*, 2020.

Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to Generate Long-term Future via Hierarchical Prediction. In *International Conference on Machine Learning*, 2017.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, 2020.

Roman V. Yampolskiy and M. S. Spellchecker. Artificial intelligence safety and cybersecurity: a timeline of ai failures, 2016.

Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency regularization for generative adversarial networks. In *International Conference on Learning Representations*, 2020.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Appendix

Consistency Regularization for Training Confidence-Calibrated Classifiers

A. Experimental details

Models and data augmentations. We use ResNet-18 and ResNet-34 (He et al., 2016) architecture for all experiments in our paper. For weak augmentation, we use random crop with padding, horizontal flip in standard classification, and Inception crop (Szegedy et al., 2015), horizontal flip in fine-grained classification. For all datasets, we use RandAugment (Cubuk et al., 2019) and Cutout (DeVries and Taylor, 2017) additionally for a strong augmentation by following (Berthelot et al., 2020).

Training detail. For CIFAR-10 and CIFAR-100 image classification, we train the model for 200 epochs with batch size 128, using stochastic gradient descent with momentum 0.9 and weight decay with 0.0001. The learning rate starts at 0.1 and is dropped by a factor of 10 at 50%, and 75%, of the training progress. For the fine-grained dataset, we follow the same training process expect with a batch size of 32, due to the smaller number of training samples.

Inference detail. Unless otherwise noted, we set the sampling number $n = 4$ for the Augment-Entropy approximation. For the data augmentation family $\mathcal{T}$ for equation (2), we choose random crop with horizontal flip.

Dataset details. For in-distribution datasets, we consider CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009), and CUB-200 (Wah et al., 2011). CIFAR-10 and CIFAR-100 consist of 50,000 training and 10,000 test images with 10 and 100 image classes, respectively. CUB-200 contains 200 species (classes) of birds, each with roughly 30 training images (total 5,994) and 30 testing images (total 5,794).

For CIFAR datasets, the out-of-distribution (OOD) datasets are as follows: SVHN (Netzer et al., 2011) consists of 26,032 test images with 10 digits, resized LSUN (Liang et al., 2018) consists of 10,000 test images of 10 different scenes, resized ImageNet (Liang et al., 2018) consists of 10,000 test images with 200 images classes from a subset of full ImageNet dataset, fixed version of LSUN* (Tack et al., 2020)[2] consists of 10,000 test images of 10 different scenes, fixed version of ImageNet*(Tack et al., 2020) consists of 10,000 test images with 30 images classes from a subset of full ImageNet dataset, Food-101 (Bossard et al., 2014) consists of 101 food categories with 101,000 images, Caltech-256 (Griffin et al., 2007) consists of object categories containing a total of 30,607 images, and MIT-67 (Quattoni and Torralba, 2009) consists of 67 Indoor categories, and a total of 15,620 images. We resized the high images into 32 by 32 resolution by using the correct resizing operation `torchvision.transforms.Resize()` by following (Tack et al., 2020), and center crop; Food-101, Caltech-256, and MIT-67 are high-resolution datasets.

For CUB-200 dataset, the considered out-of-distribution datasets are as follows: Food-101, Caltech-256, MIT-67, Places-365 (Zhou et al., 2017) with small images (256 * 256) validation set contains 36,500 images of scene categories, and Stanford Dogs (Khosla et al., 2011) which consists 120 classes, and a total of 20,580 images.

Evaluation metrics. For evaluation, we measure the two metrics that each measures (a) the effectiveness of the proposed score in distinguishing in- and out-of-distribution images, (b) the confidence calibration of softmax classifier.

- **Area under the receiver operating characteristic curve (AUROC).** Let TP, TN, FP, and FN denote true positive, true negative, false positive and false negative, respectively. The ROC curve is a graph plotting true positive rate = TP / (TP+FN) against the false positive rate = FP / (FP+TN) by varying a threshold.
- **Expected calibration error (ECE).** For a given test data $\{(x_n, y_n)\}_{n=1}^{N}$, we group the predictions into M interval bins (each of size $1/M$). Let B_m be the set of indices of samples whose prediction confidence falls into the interval $(\frac{m-1}{M}, \frac{m}{M}]$. Then, the expected calibration error (ECE) (Naeini et al., 2015; Guo et al., 2017) is follows:

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)|, \tag{7}$$

  where $\mathrm{acc}(B_m)$ is accuracy of B_m: $\mathrm{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}_{\{y_i = \arg\max_j p^{(j)}(x_i)\}}$ where $\mathbb{1}$ is indicator function and $\mathrm{conf}(B_m)$ is confidence of B_m: $\mathrm{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} q(x_i)$ where $q(x_i)$ is the confidence of data x_i. For all experiments, we set the number of bins to 20 *i.e.*, $M = 20$.

[2] use PyTorch `torchvision.transforms.Resize()` operation for correct resizing.

B. Baselines

Inference methods. we review the baseline inference methods of the out-of-distribution (OOD) detection, *i.e.*, baseline (Hendrycks and Gimpel, 2017), ODIN (Liang et al., 2018), Mahalanobis (Lee et al., 2018) and Gram matrix (Sastry and Oore, 2020). Then we show that the detection through the Gram matrix is sensitive to hyperparameter choices in some cases.

The goal of the inference method is to design a score function s upon the pre-trained classifier p_θ, so that the score of the in-distribution sample $x_{\texttt{in}}$ is higher than the score of the OOD sample $x_{\texttt{out}}$, *i.e.*$s(x_{\texttt{in}}) > s(x_{\texttt{out}})$. Some of the inference methods require an OOD validation set for the hyperparameter selection; however, we *do not assume an OOD validation set is available* at test-time. We note that such assumption is not a realistic detection scenario (see Appendix F for a more comprehensive discussion). The following is the definition of each score function and how to select hyperparameters without any OOD validation set.

- **Baseline** (Hendrycks and Gimpel, 2017)**.** They use a maximum value of softmax probablity as a score function. For a given input x the score s is as follows: $s(x) := \max_i p_\theta^{(i)}(x)$ where $p_\theta^{(i)}(x) := \frac{\exp(g_i(x))}{\sum_j \exp(g_j(x)))}$ is the prediction probability of the class i of a pre-trained classifier p_θ, and g_i is the logit value of the class i.
- **ODIN** (Liang et al., 2018)**.** They first utilize temperature scaling (at the softmax layer) and input perturbations, then use the baseline score. For a given input x, temperature T, and input perturbation scale ϵ, the score s is as follows: $s(x) := \max_i \hat{p}_\theta^{(i)}(\hat{x})$ where $\hat{p}_\theta^{(i)}(x) := \frac{\exp(g_i(x)/T)}{\sum_j \exp(g_j(x)/T))}$, and $\hat{x} = x - \epsilon \text{sign}(-\nabla_x \log \max_i \hat{p}_\theta^{(i)}(x))$. For all experiments, we fix the hyperparameter values; $T = 1000$, and $\epsilon = 0.0012$ by following the author's suggestion.
- **Mahalanobis** (Lee et al., 2018)**.** They compute the Mahalanobis distance between test sample's feature representations and the class-conditional Gaussian distribution at each layer, then they represent each sample as a vector of the Mahalanobis distances. Here, we assume to have weights of each layer's contribution α_l, and the mean and covariance of the Gaussian distribution $\hat{\mu}_{l,i}, \widehat{\Sigma}_l$ where l, i denotes the index of the layer and the class, respectively (assume that the covariance is the same for all classes). Then the detection score is as follows: $s(x) := \sum_l \alpha_l M_l$ where $M_l = \max_i -(f_l(x) - \hat{\mu}_{l,i})^T \widehat{\Sigma}_l^{-1}(f_l(x) - \hat{\mu}_{l,i})$, and f_l is the internal layer activation of index l. In this paper, we assume uniform ensemble of each layer, *i.e.*, the contribution of the layers are equal $\forall l$, $\alpha_l = 1$.
- **Gram matrix** (Sastry and Oore, 2020)**.** The high-level idea of Gram matrices is to identify inconsistency between activity patterns and predicted class. They detect anomalies in the Gram matrices by comparing each value with its respective range observed over the training data. In the paper, they introduce p-th order Gram matrix: $G_l^p = \left(f_l^p f_l^{p\top}\right)^{1/p}$ where f_l denotes the l-th layer activation for given input x (see the paper for the details). To achieve the final detection score, they compute the p-th order detection score over all $p \in P$. Following the paper, we set $P = \{1, \dots, 10\}$ for CIFAR datasets. For CUB-200 dataset, we set the hyperparameter as $P = \{1\}$.

Hyperparameter sensitivity of the Gram matrix. We observed that the Gram matrix also requires hyperparameter selection, which is highly sensitive in some cases[3]. As shown in Table 4, the Gram matrix fails to detect OOD samples (*i.e.*, AUROC lower than 50, which is a random guess) with the recommended hyperparameter (*i.e.*, $P = \{1, \dots, 10\}$), when CUB-200 is in-distribution. Note that it is an opposite observation from (Sastry and Oore, 2020) which have shown better performance on $P = \{1, \dots, 10\}$. We therefore simply fix $P = \{1\}$, for CUB-200 experiments.

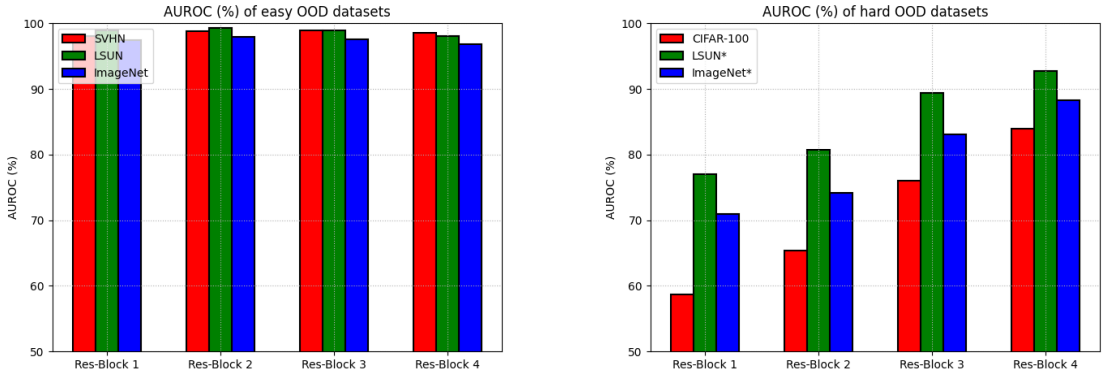
Table 4. AUROC (%) based on the hyperparameter selection of the Gram matrix. We use ResNet-18 trained on CUB-200 with standard cross-entropy loss. The final column indicates the mean AUROC across all OOD datasets.

	CUB-200 →					
Hyperparameter	Food-101	Caltech-256	MIT-67	Places-365	Dogs	Mean
$P = \{1\}$	81.04	81.74	74.64	74.00	72.15	76.71
$P = \{1, \dots, 9\}$	32.27	32.42	30.55	30.47	31.00	31.34
$P = \{1, \dots, 10\}$	0.00	0.00	0.00	0.00	0.00	0.00

Training methods. We consider two recent advanced training methods, including self-supervised learning (Hendrycks et al., 2019), and CSI (Tack et al., 2020). Self-supervised learning method train to classify the applied augmentation of a given input (3), then use the same objective as a detection score. CSI further utilize the self-supervision for OOD detection by adapting the recent advanced self-supervised contrastive learning (Chen et al., 2020).

[3]We use the official implementation from https://github.com/VectorInstitute/gram-ood-detection

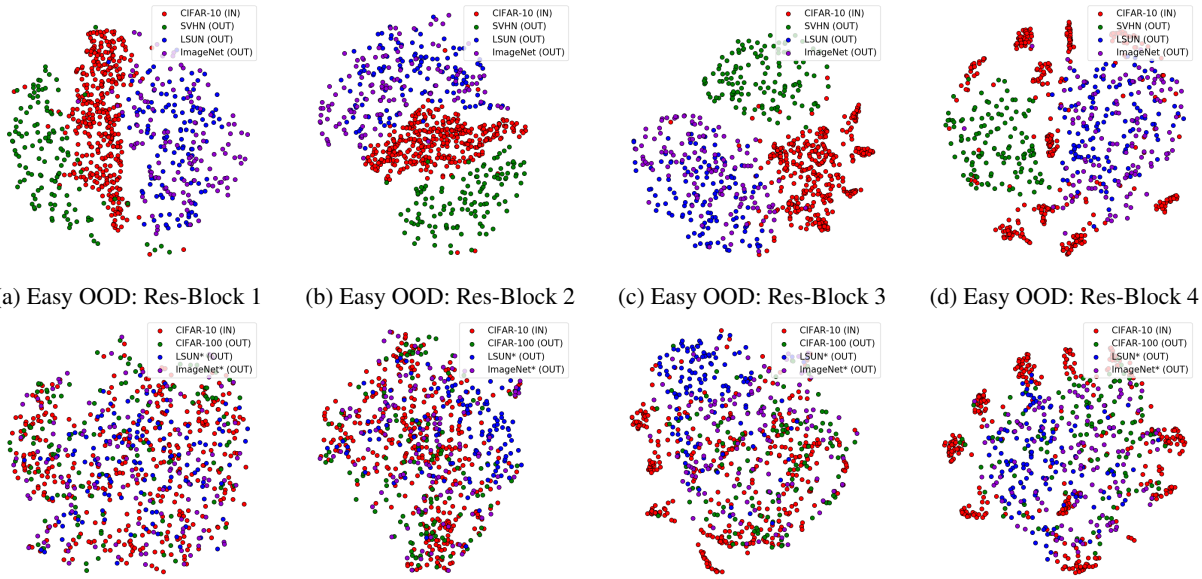
C. More discussion on out-of-distribution benchmarks



(a) Easy OOD detection result

(b) Hard OOD detection result

Figure 2. AUROC (%) of the logistic regression trained upon each residual block (*i.e.*, internal representation) of the ResNet-34. ResNet-34 is pre-trained on CIFAR-10, and for logistic regression training, we use in-and-out distribution validation sets. * denotes the fixed version dataset (*i.e.*, hard OOD dataset), and "Res-Block i" indicates the i-th residual block of the ResNet.



(a) Easy OOD: Res-Block 1

(b) Easy OOD: Res-Block 2

(c) Easy OOD: Res-Block 3

(d) Easy OOD: Res-Block 4

(e) Hard OOD: Res-Block 1

(f) Hard OOD: Res-Block 2

(g) Hard OOD: Res-Block 3

(h) Hard OOD: Res-Block 4

Figure 3. The t-SNE visualization at each residual block (*i.e.*, internal representation) of the ResNet-34, which is pre-trained on CIFAR-10. SVHN, resized LSUN and ImageNet are used as easy OOD, while CIFAR-100, fixed version of LSUN and ImageNet are used as hard OOD. * denotes the fixed version dataset (*i.e.*, hard OOD dataset), and "Res-Block i" indicates the i-th residual block of the ResNet.

In this section, we provide quantitative results and more visualization that supports our hypothesis in Section 2; easy out-of-distribution (OOD) datasets are well separated in the internal layer, while hard OOD datasets are not. To this end, we suggest to measure the separation of the in-and-out distributions by training a logistic regression to classify two distributions, on the internal represention of the classifier. Intuitively, if two distributions are well separated, then the logistic regression will achieves high classification performance. To train the regression model, we use 1,000 in-and-out distribution validation samples and evaluate on the remaining test set as a measurement of separation (we use AUROC).

For a given validation set $\mathcal{X}_{\texttt{in}}, \mathcal{X}_{\texttt{out}}$, we extract the feature map of the i-th residual block f_i as follows: $\mathcal{D}^i_{\texttt{inter}} := \{(f_i(x_{\texttt{in}}), 1) | x_{\texttt{in}} \in \mathcal{X}_{\texttt{in}}\} \bigcup \{(f_i(x_{\texttt{out}}), 0) | x_{\texttt{out}} \in \mathcal{X}_{\texttt{out}}\}$. Then for a given $(f_i(x), y) \sim \mathcal{D}^i_{\texttt{inter}}$, we train the logistic regression p_ϕ with the following loss: $\mathcal{L}_{\texttt{BCE}}(p_\phi(f_i(x)), y)$ where $\mathcal{L}_{\texttt{BCE}}$ is a binary cross-entropy loss. At the inference time, for a given test sample x, we directly use the prediction of the regression model as the detection score: $s(x) := p_\phi(f_i(x))$.

As shown in Figure 2, easy OOD samples are well separated across the entire layer, while hard OOD samples are better separated from the penultimate layer. This result supports our assumption in Section 2; some recent methodologies fail to detect hard OOD due to the internal layer usage. We also provide t-SNE visualization of OOD datasets across all residual blocks in Figure 3, which show a consistent observation.

D. More experimental results

Standard classification. We also consider the standard classification setup: we assume that in-distribution samples are from a specific multi-class dataset and train a classifier, then test on various external datasets as out-of-distribution. In addition to the the conventional OOD detection setup (Liang et al., 2018; Lee et al., 2018); where CIFAR-10 (Krizhevsky et al., 2009) is in-distribution dataset while SVHN (Netzer et al., 2011), resized LSUN and ImageNet (Liang et al., 2018) are out-of-distribution, we additionally consider the following hard OOD datasets: CIFAR-100, and fixed LSUN* and ImageNet* (Tack et al., 2020), Food-101 (Bossard et al., 2014), Caltech-256 (Griffin et al., 2007), and MIT-67 (Quattoni and Torralba, 2009). Note that we followed the same resizing operation as Tack et al. (2020) for high resolution datasets to avoid generating easy OOD datasets.

The results in Table 5 support our belief that the state-of-the-art inference methods that utilize the internal feature of the deep network (*e.g.*, Mahalanobis and Gram matrix) do not generalize in detecting hard OOD datasets (*e.g.*, LSUN*). On the other hand, our proposed method significantly outperforms the above inference methods in hard OOD detection while achieving comparable results in easy OOD detection. By combining our method with auxiliary self-supervised loss, we achieve comparable results to CSI, which is known to be one of the most strong baselines, while the training computation is much efficient (73.33% less training time[4]).

Table 5. AUROC (%) of ResNet-34 trained on CIFAR-10, and CIFAR-100. Here, we consider various *easy and hard OOD* detection scenarios. "CE", "Inter.", "HParam.", and "Cost." denote the cross-entropy, methods that utilize the internal feature of the network, methods that require hyperparameters, and the relative training cost compare to the standard cross-entropy training, respectively. The datasets with * indicate the fixed version (*i.e.*hard OOD). The final column indicates the mean AUROC across all the OOD datasets and the bold denotes results within 1% from the highest result.

(a) ResNet-34 trained on CIFAR-10 / CIFAR-100 and tested under various *easy OOD* datasets.

					CIFAR-10 →			CIFAR-100 →		
Train	Inference	Inter.	HParam.	Cost.	SVHN	LSUN	ImageNet	SVHN	LSUN	ImageNet
CE	Baseline (Hendrycks and Gimpel, 2017)	-	-	×1	88.03	87.45	84.58	85.17	71.51	71.82
CE	ODIN (Liang et al., 2018)	-	✓	×1	76.09	89.20	83.37	89.65	78.49	77.76
CE	Mahalanobis (Lee et al., 2018)	✓	✓	×1	94.98	96.74	96.35	85.33	**95.22**	**95.24**
CE	Gram matrix (Sastry and Oore, 2020)	✓	✓	×1	**99.41**	**98.85**	**98.22**	**96.97**	**95.79**	**95.12**
CSI (Tack et al., 2020)	Baseline (Hendrycks and Gimpel, 2017)	-	-	×30	98.07	97.69	**97.61**	86.25	86.13	84.85
CSI (Tack et al., 2020)	CSI-ens (Tack et al., 2020)	-	-	×30	**98.97**	**98.68**	**98.56**	88.70	88.71	87.13
CE + Self-sup (3)	Augment-Entropy (2) + Self-sup (3)	-	-	×4	**98.64**	94.99	94.48	94.12	90.90	92.09
CE + Consistency (1)	Augment-Entropy (2)	-	-	×2	97.15	96.74	95.73	86.23	84.95	84.46
Final (5)	Augment-Entropy (2) + Self-sup (3)	-	-	×8	**98.80**	96.66	97.22	95.34	90.60	92.22

(b) ResNet-34 trained on CIFAR-10 and tested under various *hard OOD* datasets.

					CIFAR-10 →						
Train	Inference	Inter.	HParam.	Cost.	LSUN*	ImageNet*	CIFAR-100	Food-101	Caltech-256	MIT-67	Mean
CE	Baseline (Hendrycks and Gimpel, 2017)	-	-	×1	89.73	88.29	86.36	86.56	85.75	89.62	87.72
CE	ODIN (Liang et al., 2018)	-	✓	×1	85.55	81.00	76.94	76.65	80.76	85.71	81.10
CE	Mahalanobis (Lee et al., 2018)	✓	✓	×1	77.76	81.74	78.74	85.76	82.67	81.41	81.35
CE	Gram matrix (Sastry and Oore, 2020)	✓	✓	×1	76.81	77.70	73.75	73.05	78.44	78.79	76.42
CSI (Tack et al., 2020)	Baseline (Hendrycks and Gimpel, 2017)	-	-	×30	**94.81**	95.22	92.67	95.63	92.18	**96.29**	94.47
CSI (Tack et al., 2020)	CSI-ens (Tack et al., 2020)	-	-	×30	**95.78**	**96.26**	**93.91**	**96.52**	93.27	**97.09**	**95.47**
CE + Self-sup (3)	Augment-Entropy (2) + Self-sup (3)	-	-	×4	86.32	89.13	85.06	93.83	87.11	91.25	88.78
CE + Consistency (1)	Augment-Entropy (2)	-	-	×2	93.84	94.07	**93.32**	92.63	**94.39**	95.35	93.93
Final (5)	Augment-Entropy (2) + Self-sup (3)	-	-	×8	94.14	**95.29**	**92.97**	**97.51**	**94.49**	**96.73**	**95.19**

[4]Without self-supervised loss, our method costs 93.33% less training time.

Test accuracy and calibration. We demonstrate the effectiveness of our method on prediction and calibration performance under various image classification datasets: CIFAR-10, CIFAR-100, and CUB-200. As shown in Table 6, the proposed consistency regularization significantly and consistently improves both prediction accuracy and confidence calibration. We note that consistent improvement in both metric is non-trivial, as the advanced baselines does not improve in all cases. We also emphasize that our proposed method is specialized for calibration, as other methods, *e.g.*, self-supervised learning (Hendrycks et al., 2019), only increase the accuracy and the calibration may not improve.

Table 6. Test accuracy (%) and ECE (%) of classifiers trained on various image classification tasks. We train ResNet-34 for CIFAR datasets and ResNet-18 for the fine-grained datasets. The arrow on the right side of the evaluation metric indicates the ascending or descending order of the value. Parentheses indicate the relative rate of ECE from the cross-entropy (CE), and bold denotes the best results.

Evaluation metric	Train method	CIFAR-10	CIFAR-100	CUB-200
Test accuracy ↑	CE	94.75	73.94	52.71
	CE + Self-sup (3)	96.72	77.47	60.98
	CSI (Tack et al., 2020)	95.58	77.94	43.34
	Ours, $\mathcal{L}_{\texttt{final}}$(5)	**96.79 (+2.15%)**	**80.78 (+9.25%)**	**61.63 (+16.92%)**
ECE ↓	CE	3.64	11.81	20.35
	CE + Self-sup (3)	2.65	11.88	17.49
	CSI (Tack et al., 2020)	2.37	8.56	26.21
	Ours, $\mathcal{L}_{\texttt{final}}$(5)	**1.51 (-58.51%)**	**8.17 (-30.82%)**	**6.78 (-66.68%)**

E. Ablation study

Component analysis. In Table 7, we assess the individual effects of each component of our final training objective and adding consistency loss improves all metrics such as test accuracy, calibration error and AUROC. Even though adding each component might not always improve the all metrics, we observe that adding components in the training objective is beneficial in most cases. Intriguingly, the consistency regularization loss and self-supervised loss improves both classification accuracy and the softmax calibration, the unlabeled loss is rather more specified to the model calibration.

Table 7. Ablation study on each component of our proposed training loss. We measure test accuracy (%), ECE (%), and AUROC (%) where CIFAR-100 is in-distribution, and CIFAR-10 is out-of-distribution. For all objective, we used the same score (*i.e.*, baseline (Hendrycks and Gimpel, 2017)) and network architecture (*i.e.*, ResNet-34). No checkmark and full checkmark denote the standard cross-entropy training, and the extended version (5), respectively. The best results are denoted in bold.

Consistency, $\mathcal{L}_{\texttt{con}}$ (1)	Self-sup, $\mathcal{L}_{\texttt{self}}$ (3)	Unlabeled, $\mathcal{L}_{\texttt{unlabel}}$ (4)	Test acc.	ECE	AUROC
-	-	-	73.94	11.81	76.2
✓	-	-	77.81	10.73	77.7
-	✓	-	77.47	11.88	76.94
✓	✓	-	80.65	9.87	**78.75**
✓	✓	✓	**80.78**	**8.17**	78.55

Effect of the sampling number. In Table 8, we investigate an effect of the sampling number used for approximating Augment-Entropy (2). We note that the proposed Augment-Entropy indeed improve the OOD detection. Surprisingly, we found that the sampling number of 4 is sufficient. The reason is that our consistency regularization already have forced the augmentation to be concentrated. As a result, we can effectively approximate the expectation with a small number of samples. Therefore, our method can be effectively used at a small cost.

Table 8. Ablation study on the effect of sampling number n for Augment-Entropy approximation. We measure AUROC (%) where CIFAR-100 is in-distribution, and CIFAR-10 is out-of-distribution, and the network is ResNet-34.

n	1	2	4	8	16
AUROC	78.77	79.61	80.20	80.25	80.35

F. Discussion on OOD validation set

We give an empirical evidence that one should avoid using an out-of-distribution validation set. In such a detection scenario, we observed that the problem can be solved by a straightforward method, and it is comparable with most of the baseline works. The high-level idea is to learn a logistic regression on top of the pre-trained representation by utilizing the OOD validation set. Consider a depth-d pre-trained network which output for given input x is $f(x; W_{1:d}) = W_d\sigma(\dots\sigma(W_1 x))$ where W_i is a i^{th} layer weight and σ is an activation function. We then aggregate (*i.e.*, concatenate) all the internal feature and use as the input feature for logistic regression: $x_{\texttt{input}} = \texttt{aggregate}(\{f(x; W_{1:i})\}_{i=1}^{d})$. Finally, we train a logistic regression function with the extracted features of in-and-out validation samples. For training the logistic regression, we use the same pre-trained network (*i.e.*, ResNet-34) and validation set from the previous work (Lee et al., 2018).[5]

The result in Table 9 shows that the simple regression achieves very high AUROC performance (over 96%) for popular OOD setups, even comparable to the state-of-the-art results when the OOD validation set is used. Note that the logistic regression achieves the result without even careful input preprocessing (Liang et al., 2018; Lee et al., 2018) used in the prior works.

Table 9. AUROC (%) value of detection under logistic regression trained using an OOD validation set. We train the logistic regression upon the same pre-trained network (*i.e.*, ResNet-34) and validation set from the previous work (Lee et al., 2018).

In-distribution	OOD	AUROC
CIFAR-10	SVHN	98.9
	ImageNet	99.0
	LSUN	99.4
CIFAR-100	SVHN	97.8
	ImageNet	96.3
	LSUN	98.4

[5]The pre-trained network and validation set are available at https://github.com/pokaxpoka/deep_Mahalanobis_detector.