This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

RankOOD - Class Ranking-based Out-of-Distribution Detection

Dishanika Denipitiyage[1] Naveen Karunanayake[1] Suranga Seneviratne[1] Sanjay Chawla[2]
{dishanika.denipitiyage, naveen.karunanayake, suranga.seneviratne}@sydney.edu.au
schawla@hbku.edu.qa
[1] The University of Sydney [2] Qatar Computing Research Institute, HBKU

Abstract

We propose RankOOD, a rank-based Out-of-Distribution (OOD) detection approach based on training a model with the Placket-Luce loss, which is now extensively used for preference alignment tasks in foundational models. Our approach is based on the insight that with a deep learning model trained using the Cross Entropy Loss, in-distribution (ID) class prediction induces a ranking pattern for each ID class prediction. The RankOOD framework formalizes the insight by first extracting a rank list for each class using an initial classifier and then uses another round of training with the Plackett-Luce loss, where the class rank, a fixed permutation for each class, is the predicted variable. An OOD example may get assigned with high probability to an ID example, but the probability of it respecting the ranking classification is likely to be small. RankOOD achieves SOTA performance on the near-ODD TinyImageNet evaluation benchmark, reducing FPR95 by 4.3%.

1. Introduction

Despite their tremendous success, deployment of Deep Neural Networks (DNNs) in critical applications remains restricted due to the out-of-distribution (OOD) problem. For example, a vision classifier installed in an autonomous driving vehicle may make an inaccurate but overconfident prediction for an object class not seen in the training data. Accurate OOD detection continues to remain a fundamental unresolved problem not only in computer vision, but also in all machine learning [9, 25, 37].

Existing OOD detection approaches can be broadly divided into two categories: post-hoc methods and training-based methods. Post-hoc methods [7, 9, 19, 27, 32] operate on pretrained models and extract OOD-related signals from their outputs or internal representations, offering simplicity and compatibility with existing networks. In contrast, training-based methods modify the learning process to improve the separability between in-distribution (ID) and OOD data. Among these, outlier-exposure techniques [10, 35, 36, 38] explicitly use auxiliary outlier datasets during training, while training without outliers methods enhance OOD robustness implicitly through regularization or objective design. Although post-hoc approaches are lightweight and maintain ID performance, training-based approaches typically achieve stronger OOD detection, often at the cost of reduced classification accuracy on ID samples.

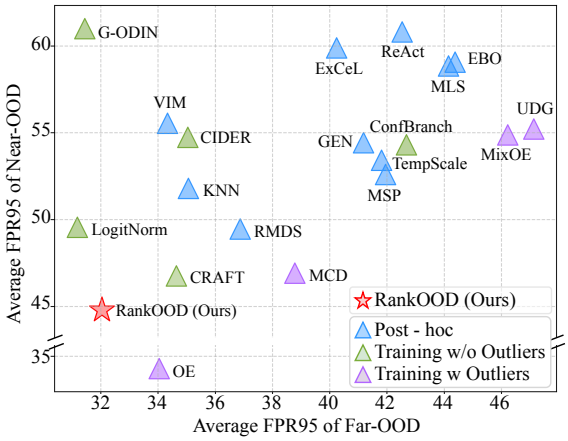


Figure 1. The performance comparison of average FPR95 on Far-OOD (x-axis) and Near-OOD (y-axis). *RankOOD outperforms all post-hoc and training methods without outliers. Compared to training methods with outliers, its performance is only second to the Outlier Exposure (OE) method.*

Recently, a new line of work is emerging on addressing the OOD problem, which is post-hoc and includes elements of post-training. The key insight is the following:

> A DNN classifier trained for single-class prediction often naturally induces a rank order across classes. An OOD example may be overconfidently assigned to a class label but is unlikely to respect the associated rank order. Thus, if we can train a classifier to strengthen the rank order, then we can achieve both accurate in-distribution prediction and OOD detection.

Several works leverage class ranking patterns as a discriminative signal for OOD detection [1, 14, 15]. For

instance, CRAFT [15] fine-tunes pre-trained models to learn class-specific ranking distributions among prediction scores. CRAFT demonstrates that ID samples exhibit more deterministic inter-class ranking relationships, while OOD samples disrupt these patterns. By modeling these relationships via class-dependent probability mass functions (PMFs) and measuring their divergence during inference, CRAFT achieves strong OOD performance. However, this fine-tuning process introduces architectural modifications and ignores the relative ordering of the class ranks.

In this work, we propose RankOOD, a rank-based, listwise OOD detection framework that eliminates the need for fine-tuning or a separate network while preserving the discriminative power of ranking structures. Unlike CRAFT, which models each class ranking as a 2D C×C PMF matrix (where C is the number of classes), RankOOD directly operates on the raw logits of a pre-trained network and orders them according to predefined rank labels. We employ a Listwise Maximum Likelihood Estimation (ListMLE) [34] objective to learn the rank structure. Unlike pointwise or pairwise objectives, ListMLE treats the entire output list jointly and assigns a probability to each permutation via the Plackett–Luce model [21], maximizing the likelihood of the observed ranking. This allows the model to capture relative ordering dependencies among classes, reflecting their competitive nature in the final decision. Conceptually, our method captures global listwise consistency, modeling how class scores interact within a shared ranking space. This perspective not only simplifies the training pipeline, since no additional fine-tuning or outlier exposure is required, but also provides a canonical ranking structure for OOD detection. We summarize the following contributions,

- We propose RankOOD, a novel method that detects OOD inputs by learning and analyzing class-wise rank order patterns directly from model outputs without fine-tuning or auxiliary outliers.
- We introduce the novel use of a ListMLE objective derived from the Plackett–Luce (PL) model [21], to model entire output rankings for OOD detection, effectively capturing vital inter-class dependencies ignored by prior methods.
- We validate RankOOD by comparing 34 existing methods conducted in the OpenOOD environment [39]. Our method consistently ranks within the top two in near-OOD detection and the top three for far-OOD setting, demonstrating the strong consistency among the evaluated methods. Notably, on the TinyImageNet near-OOD, RankOOD achieves SOTA performance, reducing FPR95 by 4.3% relative to the strongest baseline.

2. Related Work

OOD detection methods for image classifiers are primarily grouped into two categories: i) post-hoc inference methods [9, 18, 19] and ii) training-based methods [10, 11, 33].

Post-hoc inference methods: These methods utilize a standard classifier (usually trained with cross-entropy) and compute an OOD score from its outputs or intermediate features. A simple baseline is the Maximum Softmax Probability (MSP) detector, which uses the largest softmax score as the anomaly score [9]. Other classic scores include the Mahalanobis distance [18], which measures deviations from class-conditional feature statistics, and the energy score [19], defined as the negative log-sum-exp of the logits. More recent methods aim to further improve the separability between ID and OOD samples by refining DNN outputs or activations through techniques such as rectification [27], shaping [6], and sparsification [28].

Training-based methods: Following the OpenOOD taxonomy [39], training-based methods can be further grouped into methods without outliers [13, 22, 33] and with outliers [10, 36, 38]. *Training methods without outliers* modify the learning objective using only ID data to improve calibration and robustness. RotPred [11] introduces an auxiliary rotation prediction task, leveraging self-supervision to encourage feature diversity that aids OOD detection. CSI [29] applies contrastive self-supervised learning to cluster ID samples tightly in embedding space. LogitNorm [33] enforces a fixed norm on the output logits during training, preventing overconfident predictions and substantially improving post-hoc OOD separability. These methods offer strong performance without relying on external data, making them practical for closed-set training scenarios.

Conversely, when auxiliary outlier data are available, models can explicitly learn to assign low confidence to them (i.e., *training methods with outliers*). The outlier exposure (OE) method [10] introduces a loss term that penalizes overconfident predictions on a diverse auxiliary dataset, encouraging uniform output distributions for OOD samples. Building on OE, MixOE [38] interpolates ID and OOD samples to generate mixed examples that regularize model confidence. Other variants, such as MCD [36] and UDG [35], leverage ensemble disagreement or unsupervised clustering to synthesize pseudo-OOD data.

Each category of methods has its own strengths and trade-offs. Post-hoc approaches are simple to apply and require no retraining, but their effectiveness depends heavily on the quality and calibration of the underlying model. Training-based methods without outliers tend to be more robust and data-efficient, though they may struggle when confronted with unfamiliar, real-world anomalies. Outlier-assisted methods often achieve the strongest results but risk overfitting to the seen outlier data and rely on access to sufficiently diverse and representative outlier examples, which may not always be practical.

Class rank-based OOD detection A recent line of research has leveraged class-rank information to improve

OOD detection. For example, ExCeL [14] computes a post-hoc OOD score by combining the maximum logit with a *class rank signature* that captures the probability of each class appearing in subsequent ranks. Extending this idea, CRAFT [15] enhances OOD detection by fine-tuning a pre-trained classifier to sharpen its learned class-rank patterns, which are modeled as PMFs. *Our work, RankOOD further advances this direction by optimizing class-wise rank structures under the Plackett–Luce formulation [21], using standard vision backbones without architectural changes.*

3. Methodology

In this section, we present the overall methodology of RankOOD, which consists of three main steps: i) using a pre-trained model on ID data, we derive a canonical class ranking for each class by solving a rank assignment problem (Sec. 3.1); ii) we then train a new classifier on the ID data using the canonical class rankings as ground truth and the ListMLE loss, referred to as RankOOD-T, the training process of our framework (Sec. 3.2); and iii) during inference, we compute an OOD score (termed RankOOD-S) based on the predicted ranks and their deviation from the canonical rankings, under the premise that OOD samples deviate more from the canonical class ranking patterns (Sec. 3.3).

3.1. Finding Canonical Class Ranks

Using a pre-trained model on in-distribution train data, we first compute a *Rank Probability Matrix* (RPM) [15] for each class $c \in C$, which comprises the probability mass functions (PMFs) across all rank positions, estimated from ID samples that are correctly predicted as class c. Specifically, for each class c, an element $p^c_{i,j}$ within the RPM, $P^c \in \mathbb{R}^{C\times K}$ denotes the probability that class i appears at the $j^{th} \in K$ rank when the input is classified as class c. Thus, each column of the matrix P^c_j represents a PMF over ID classes corresponding to a particular rank position j.

To obtain a consistent canonical class ranking for each top-rank class, we solve an Integer Linear Programming (ILP) problem. That is, we formulate a 0-1 integer linear program that selects one class per rank, ensuring uniqueness and maximizing the total assignment probability. For each class, this yields a consistent ranking that best reflects the trained preference structure.

Given the class c, and its rank probability matrix, $P^c \in \mathbb{R}^{C\times K}$, we introduce binary decision variables,

$$x^c_{i,j} = \begin{cases} 1, & \text{if class } i \text{ is assigned to rank } j, \\ 0, & \text{otherwise.} \end{cases}$$

We compute the consensus ranking by solving the following 0-1 integer linear program:

$$\max_x \sum_{i=1}^{C}\sum_{j=1}^{K} x^c_{i,j} p_{i,j} \tag{1}$$

subject to

$$\sum_{i=1}^{C} x^c_{i,j} = 1 \forall j \in [1,K], \quad \sum_{j=1}^{K} x^c_{i,j} \leq 1 \forall i \in [1,C]$$

This yields a valid ranking permutation that maximizes the joint probability of the K ranking structure under the model. Note that the first constraint ensures only one class is selected per rank, and the second constraint ensures that a class is selected at most once, across all considered ranks.

Example: Consider this worked example in a four-class classification problem as shown in Table 1. We consider the 100 correctly classified samples of Class 2.

Since we are considering only 100 correctly classifying samples of Class 2, the frequency f^2_{ij} for (2,0), i.e., the number of samples where Class 2 is in the 0^{th} rank is 100. This also means, there are no samples with any other class in rank 0, i.e., $f^2_{i0}\ \forall\ i\backslash\{2\}$ is 0. Also means that Class 2 can not appear in any other position than rank-0, i.e., $f^2_{2j}\ \forall\ j\backslash\{0\}$ is 0.

Consider the other f^2_{ij} values as filled in the table. Note that the j^{th} column sum, i.e., $\sum_{j=1}^{4} f^2_{ij}$ is 100. Next, the relative frequency can be converted to probabilities, p^2_{ij}, as shown inside brackets in the table. The shaded cells of this table, ignoring the trivial row and column for class 2, form the RPM $\in 4\times 4$ for Class 2.

Note that in this example, for Class 2, rank 1 is dominated by Class 1, and rank 2 is dominated by Class 3, and rank 3 is dominated by Class 4. Though the canonical class is clear cut in this example, in practice, as the number of classes increases, RPMs get noisier, leading to ties between classes in some ranks. Our ILP solution ensures that the best representative ranking order is selected for each class.

		Rank (j)			
		0	1	2	3
Class (i)	1	0	80 (0.80)	15 (0.15)	0 (0.00)
	2	100	0	0	0
	3	0	10 (0.10)	75 (0.75)	1 (0.01)
	4	0	10 (0.10)	10 (0.10)	99 (0.99)

Table 1. An example RPM for Class 2, in a four class classification problem

3.2. Ordered Preference Learning

Next, leveraging the ranking information generated through ILP, we train a model using the hybrid objective combining cross-entropy (CE) loss and Listwise Maximum Likelihood Estimation (ListMLE) [34]. While CE encourages the model to predict the correct top class, ListMLE enforces consistency across the entire ordered label sequence

by maximizing the likelihood of the observed ranking under the Plackett–Luce formulation [21]. The ListMLE loss is defined as:

$$\mathcal{L}_{\text{ListMLE}} = -\sum_{i=0}^{K-1}(l_{\pi_i} - log(\sum_{j=i}^{K-1}\exp(l_{\pi_j}))) \quad (2)$$

equivalently expressed under the Plackett–Luce model as

$$\mathcal{P}(\pi|l) = \prod_{i=0}^{K-1}\frac{\exp(l_{\pi_i})}{\sum_{j=i}^{K-1}\exp(l_{\pi_j})}, \quad \mathcal{L}_{\text{ListMLE}} = -log[\mathcal{P}(\pi|l)] \quad (3)$$

Here, $\pi = (\pi_0, \pi_1, \ldots, \pi_{K-1})$ denote the ground-truth ranking where π_i is the class index assigned to rank i and l_{π_i} denote the logits of the class at rank i produced by the model. The Eq. 3 defines a parameterized exponential probability distribution over all the permutations given the predicted result by the model $\mathcal{F}_\theta$, and defines the loss function (Eq. 2) as the negative log likelihood of the ground truth ranks. This formulation enforces relative ordering $(l_{\pi_0} > \ldots > l_{\pi_i} > \ldots > l_{\pi_{K-1}})$ consistency across the entire ranked list rather than optimizing only the top-1 classification decision. Unlike conventional CE loss, which encourages only the correct label to have maximal score, ListMLE optimizes the full permutation likelihood, ensuring a coherent and structured logit hierarchy. This property makes ListMLE particularly suitable when the model must preserve rich class-relation structure and when OOD detection depends on stable, interpretable score ordering beyond the top prediction. We empirically show that it is not necessary to train the model on the entire rank sequence when only a subset of ranks is sufficient for the downstream OOD score and we selected top-k ranks and lowest-k ranks. However, training on a subset of ranks alone does not constrain the absolute ordering of unsupervised middle positions, nor does it guarantee that the l_{π_0} is globally maximal among all C logits. Therefore, to ensure that the predicted top class is indeed the argmax of the full logit vector, we therefore complement the ListMLE loss with a cross-entropy term,

$$\mathcal{L}_{RankOOD-T} = \mathcal{L}_{CE} + \alpha\mathcal{L}_{\text{ListMLE}} \quad (4)$$

The hyperparameter α balances the trade-off between the two objectives.

3.3. Detecting OOD Samples

To effectively distinguish ID samples from OOD inputs, we leverage the structure of rank-based logits learned by our RankOOD-T model. First, we derive class-dependent logit threshold profiles from confident training samples, capturing characteristic logit decay across ranks for correctly classified samples. Second, at test time, we compute a RankOOD-S that penalizes deviations from both the expected ranking order and the reference logit thresholds.

Logit threshold Profile - Ref**:** We construct a reference threshold profile for each class based on the logit values of correctly predicted training samples at rank-0. Among these, we further retain only those samples that satisfy the condition that at least N rank positions yield correct predictions, where N is chosen such that each class has at least one qualifying training sample. For each rank position i, we compute the class-specific reference logit threshold, Ref_i^c as the empirical 95^{th}-percentile of logits. In Sec. 5.4 we explain the percentile selection using the validation set provided by the OpenOOD benchmark [39].

RankOOD-S: Let x denote the logit vector for a given test input, we first determine its predicted class label $\hat{c}$ (i.e., the class related to the highest logit value in x) and obtain the expected ranking order $\pi^{\hat{c}}$ for the predicted class $\hat{c}$. Let $\bar{\pi}$ denote the ranking predicted by the input sample. For each rank position i, we introduce a penalty term if the model misorders the rank, $\bar{\pi}_i \neq \pi_i^{\hat{c}}$. Since the Plackett–Luce objective couples rank i with all preceding ranks $[0, i]$, an incorrect rank at position i contributes to a deviation in logits in previous ranks. Therefore, we define a cumulative margin penalty on logits in each expected rank i:

$$\delta_{\pi_i^{\hat{c}}} = \gamma^r, \quad r = \sum_{j=i}^{K-1} 1[\pi_j^{\hat{c}} \neq \bar{\pi}_j] \quad (5)$$

where $\gamma \geq 1$. Finally, the RankOOD-S measures the weighted deviation of penalized logits from the class-specific reference threshold profile:

$$\text{RankOOD-S} = \sum_{i=0}^{K-1} w_i \log\left(\text{softmax}(\mathbf{u})\right)_i \quad (6)$$

$$\text{where} \quad u_i = \frac{x_{\pi_i^{\hat{c}}}}{\delta_{\pi_i^{\hat{c}}}} - Ref_i^{\hat{c}} \quad \forall i \in \{0, \ldots, K-1\}$$

where w_i is learned via linear regression on validation data to maximize separation between ID and OOD samples. Intuitively, ID samples are expected to exhibit a high rank-0 logit value followed by a monotonically decreasing logit sequence that maintains the logit ordering structure across ranks. Since ListMLE in RankOOD-T loss enforces a sequential scoring dependency, a single mid-rank violation reveals inconsistency in the entire confidence trajectory, making the method inherently sensitive to OOD distortions. An example of RankOOD-S is provided in the Appendix A.5.

4. Experiments

Datasets: We use CIFAR-10/100 [16] and ImageNet-200 (a.k.a., TinyImageNet) [17] as ID datasets for our experiments. Following the OpenOOD benchmark [39], we evaluate each ID dataset against both near-OOD and far-OOD test datasets. Specifically, for CIFAR-10, the near-OOD

datasets include CIFAR-100 and TinyImageNet, while for CIFAR-100, CIFAR-10 and TinyImageNet are used as near-OOD. For both CIFAR-10 and CIFAR-100, the far-OOD group comprises MNIST [4], SVHN [23], Textures [3], and Places365 [40]. For ImageNet-200, SSB-hard [31] and NINCO [2] serve as near-OOD datasets, whereas iNaturalist [30], Textures [3], and OpenImage-O [32] form the far-OOD group. To ensure consistency, we adopt the same training, validation, and testing splits provided by the OpenOOD benchmark.

Evaluation Metrics: Similar to previous work, we use two standard metrics to evaluate OOD detection performance: (1) FPR95, the false positive rate for OOD samples when the true positive rate for ID samples is 95%; and (2) AUROC, the area under the receiver operating characteristic curve, which measures the detector's ability to distinguish between ID and OOD samples across all thresholds. We report the mean and standard deviation of these metrics over three independent runs with different random seeds.

Baselines: We compare RankOOD against a comprehensive set of 34 OOD detection methods. We categorize the baselines into three major categories based on the inference methods and the training methods: post-hoc methods, training methods without outliers, and training methods with outliers. Among post-hoc methods, we include classical approaches such as MSP [9] and Energy-based OOD detection (EBO) [19], as well as more recent techniques like ReAct [27], ASH [6], and GEN [20]. Training-based methods without auxiliary outlier data include ConfBranch [5], G-ODIN [13], and LogitNorm [33]. Conversely, methods that utilize auxiliary outlier data for training include Outlier Exposure (OE) [10], MCD [36], UDG [35], and MixOE [38]. Finally, we compare our method against two existing rank-based methods, ExCeL [14] and CRAFT [15].

Implementation Details: For all experiments, we use pre-trained ResNet-18 models [8] as the backbone. For both CIFAR datasets, each model is trained for 500 epochs, while the TinyImageNet model is trained for 300 epochs using RankOOD-T loss. All models use the SGD optimizer with a momentum of 0.9, an initial learning rate of 0.1, and a cosine annealing decay schedule. In our method, we perform hyperparameter sweeps over the loss weighting parameter α in Eq. 4 as well as over the reference logit threshold, Ref. We set α to $0.8, 1.0$, and 0.5 for CIFAR-10, CIFAR-100, and TinyImageNet, respectively, and select the 95^{th} percentile of the logit distribution as the reference logit threshold for all datasets. The CIFAR-10 network is trained using all 10 ranks, whereas the CIFAR-100 and TinyImageNet models are trained using the top 10 and bottom 10 ranks generated through the ILP procedure. Hyperparameter search details are provided in Sec. 5.4. Additional details on GPU hours, ILP complexity, and ILP runtime are provided in the Appendix A.4.

5. Results

We summarize near-OOD results in Tab. 2 and far-OOD results in Tab. 3 on CIFAR-10/100 and TinyImageNet. For brevity, we present average near- and far-OOD performance across all benchmarks for each ID dataset (Sec. 5.1), as well as the overall OOD detection performance. Complete results for all 34 methods, including per-dataset performance, are provided in the Appendix A.1 and A.2.

5.1. Comparison with SOTA Methods

Across the average performance over all benchmarks, RankOOD consistently ranks among the top three methods, achieving the second-best near-OOD (**cf.** Tab. 2) in terms of both AUROC and FPR95. RankOOD is also the third-best far-OOD (**cf.** Tab. 3) performance. Similarly, RankOOD outperforms all prior rank-based approaches, CRAFT [15] and ExCel [14] in both near- and far-OOD settings, achieving an average FPR95 reduction of 7.51% for far-OOD and 4.21% for near-OOD, while outperforming both CIFAR-100 and TinyImageNet datasets and lies on-par with CIFAR-10 performance.

In the near-OOD setting (**cf.** Tab. 2), the only method surpassing RankOOD is OE [10], which benefits from access to outlier samples during training—an assumption RankOOD does not make. Notably, in the near-OOD setting, RankOOD achieves state-of-the-art performance on TinyImageNet, improving AUROC by 0.50% and reducing FPR95 by 4.3% compared to the strongest baseline. This highlights the value of incorporating semantic rank structure when detecting OOD samples in high-cardinality label spaces. Furthermore, among methods that do not utilize outliers, RankOOD obtains the best FPR95 on CIFAR-100, outperforming GEN [20] by 3.36%.

As shown in Tab. 3, despite being on par with top five methods for CIFAR-10, RankOOD ranks second on CIFAR-100 with 83.63% AUROC and 47.44% FPR95, trailing only G-ODIN [13]. Nonetheless, RankOOD outperforms G-ODIN by 8.12% in TinyImageNet FPR95, and G-ODIN performs worst in near-OOD setting for all the datasets. Moreover, in far-OOD setting, RankOOD outperforms all outlier-exposure-based methods on CIFAR-100 and TinyImageNet across all metrics.

5.2. Qualitative Study of Rank Distributions

To analyze how rank-based training influences logit distributions, we examine the class-conditional logit distributions across different rank positions on CIFAR-100 under (a) standard CE training and (b) RankOOD-T. For a given predicted class, we sample logits corresponding to five rank positions (rank-0, three intermediate ranks, and rank-99) and measure their evolution across training epochs. Next, we compare these distributions for ID and OOD samples.

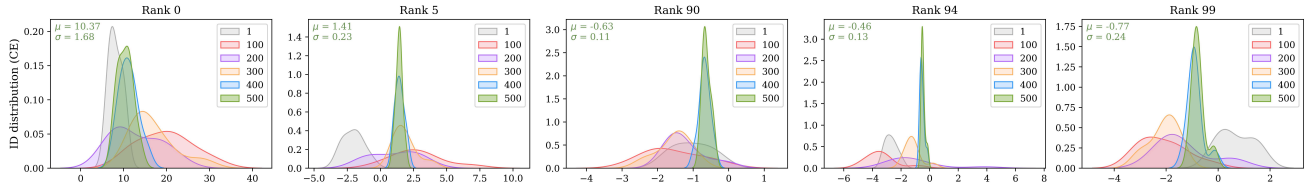
Table 2. Performance comparison in *near-OOD detection*. For each column, the top five methods are marked in **bold**.

Method	CIFAR-10		CIFAR-100		ImageNet-200		Average	
	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓
Post-hoc inference methods								
MSP [9]	88.03 ± 0.25	48.17 ± 3.92	80.27 ± 0.11	**54.80 ± 0.33**	83.34 ± 0.06	54.82 ± 0.35	83.88	52.60
TempScale [7]	88.09 ± 0.31	50.96 ± 4.32	80.90 ± 0.07	**54.49 ± 0.48**	**83.69 ± 0.04**	54.82 ± 0.23	84.23	53.42
RMDS [24]	89.80 ± 0.28	38.89 ± 2.39	80.15 ± 0.11	55.46 ± 0.41	82.57 ± 0.25	**54.02 ± 0.58**	84.17	**49.46**
EBO [19]	87.58 ± 0.46	61.34 ± 4.63	**80.91 ± 0.08**	55.62 ± 0.61	82.50 ± 0.05	60.24 ± 0.57	83.66	59.07
ReAct [27]	87.11 ± 0.61	63.56 ± 7.33	80.77 ± 0.05	56.39 ± 0.34	81.87 ± 0.98	62.49 ± 2.19	83.25	60.81
MLS [12]	87.52 ± 0.47	61.32 ± 4.62	**81.05 ± 0.07**	55.47 ± 0.66	82.90 ± 0.04	59.76 ± 0.59	83.82	58.85
VIM [32]	88.68 ± 0.28	44.84 ± 2.31	74.98 ± 0.13	62.63 ± 0.27	78.68 ± 0.24	59.19 ± 0.71	80.78	55.55
KNN [26]	90.64 ± 0.20	34.01 ± 0.38	80.18 ± 0.15	61.22 ± 0.14	81.57 ± 0.17	60.18 ± 0.52	84.13	51.80
ASH [6]	75.27 ± 1.04	86.78 ± 1.82	78.20 ± 0.15	65.71 ± 0.24	82.38 ± 0.19	64.89 ± 0.90	78.62	72.46
GEN [20]	88.20 ± 0.30	53.67 ± 3.14	**81.31 ± 0.08**	**54.42 ± 0.33**	**83.68 ± 0.06**	55.20 ± 0.20	**84.40**	54.43
ExCeL [14]	86.89 ± 0.23	66.55 ± 0.43	80.70 ± 0.06	55.21 ± 0.56	82.40 ± 0.04	57.90 ± 0.40	83.33	59.89
Training methods without outliers								
RankOOD (ours)	90.21 ± 0.41	**31.72 ± 0.67**	80.67 ± 0.40	**52.59 ± 0.75**	**85.30 ± 0.18**	**50.05 ± 0.16**	**85.39**	**44.79**
CRAFT [15]	**91.11 ± 0.04**	**31.94 ± 1.41**	80.90 ± 0.33	53.73 ± 0.62	**83.65 ± 0.41**	**54.62 ± 0.57**	**85.22**	46.76
ConfBranch [5]	89.84 ± 0.24	31.28 ± 0.66	71.60 ± 0.62	70.21 ± 0.83	79.10 ± 0.24	61.44 ± 0.34	80.18	54.31
G-ODIN [13]	89.12 ± 0.57	45.54 ± 2.52	77.15 ± 0.28	67.58 ± 0.98	77.28 ± 0.10	69.87 ± 0.46	81.18	61.00
LogitNorm [33]	**92.33 ± 0.08**	**29.34 ± 0.81**	78.47 ± 0.31	62.89 ± 0.57	82.66 ± 0.15	56.46 ± 0.37	**84.49**	**49.56**
CIDER [22]	**90.71 ± 0.16**	32.11 ± 0.94	73.10 ± 0.39	72.02 ± 0.31	80.58 ± 1.75	60.10 ± 0.73	81.46	54.74
Training methods with outliers								
OE [10]	**94.82 ± 0.21**	**19.84 ± 0.95**	**88.30 ± 0.10**	**30.73 ± 0.11**	**84.84 ± 0.16**	**52.30 ± 0.67**	**89.32**	**34.29**
MCD [36]	**91.03 ± 0.12**	**30.17 ± 0.06**	77.07 ± 0.32	55.88 ± 0.85	83.62 ± 0.09	**54.71 ± 0.83**	83.91	**46.92**
UDG [35]	89.91 ± 0.25	35.34 ± 0.95	78.02 ± 0.10	61.42 ± 0.48	74.30 ± 1.63	68.89 ± 1.72	80.74	55.22
MixOE [38]	88.73 ± 0.82	51.45 ± 7.78	**80.95 ± 0.20**	55.22 ± 0.49	82.62 ± 0.03	57.97 ± 0.40	84.10	54.88

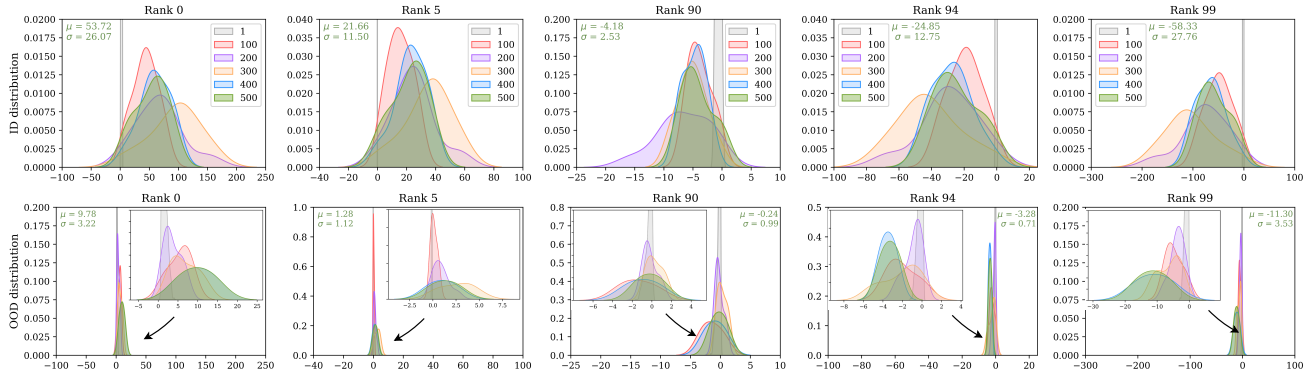
Table 3. Performance comparison in *far-OOD detection*. For each column, the top five methods are marked in **bold**.

Method	CIFAR-10		CIFAR-100		ImageNet-200		Average	
	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓
Post-hoc inference methods								
MSP [9]	90.73 ± 0.43	31.72 ± 1.84	77.76 ± 0.44	58.70 ± 1.06	90.13 ± 0.09	35.43 ± 0.38	86.21	41.95
TempScale [7]	90.97 ± 0.52	33.48 ± 2.39	78.74 ± 0.51	57.94 ± 1.14	90.82 ± 0.09	34.00 ± 0.37	86.84	41.81
RMDS [24]	92.20 ± 0.21	25.35 ± 0.73	**82.92 ± 0.42**	52.81 ± 0.63	88.06 ± 0.34	32.45 ± 0.79	87.73	36.87
EBO [19]	91.21 ± 0.92	41.69 ± 5.32	79.77 ± 0.61	56.59 ± 1.38	90.86 ± 0.21	34.86 ± 1.30	87.28	44.38
ReAct [27]	90.42 ± 1.41	44.90 ± 8.37	80.39 ± 0.49	54.20 ± 1.56	**92.31 ± 0.56**	28.50 ± 0.95	87.71	42.53
MLS [12]	91.10 ± 0.89	41.68 ± 5.27	79.67 ± 0.57	56.73 ± 1.33	91.11 ± 0.19	34.03 ± 1.21	87.29	44.15
VIM [32]	93.48 ± 0.24	25.05 ± 0.52	81.70 ± 0.62	**50.74 ± 1.00**	91.26 ± 0.19	**27.20 ± 0.30**	88.81	**34.33**
KNN [26]	92.96 ± 0.14	24.27 ± 0.40	**82.40 ± 0.17**	53.65 ± 0.28	**93.16 ± 0.22**	**27.27 ± 0.75**	**89.51**	35.06
ASH [6]	78.49 ± 2.58	79.03 ± 4.22	80.58 ± 0.66	59.20 ± 2.46	**93.90 ± 0.27**	**27.29 ± 1.12**	84.32	55.17
GEN [20]	91.35 ± 0.69	34.73 ± 1.58	79.68 ± 0.75	56.71 ± 1.59	91.36 ± 0.10	32.10 ± 0.59	87.46	41.18
ExCeL [14]	91.69 ± 0.18	40.03 ± 0.84	**82.04 ± 0.90**	**52.24 ± 1.90**	91.97 ± 0.27	28.45 ± 0.80	88.57	40.24
Training methods without outliers								
RankOOD (ours)	93.19 ± 0.84	20.96 ± 2.55	**83.63 ± 1.06**	**47.44 ± 0.80**	92.14 ± 0.20	**27.73 ± 0.33**	**89.65**	**32.04**
CRAFT [15]	93.94 ± 0.20	**19.40 ± 0.88**	82.03 ± 0.34	**51.86 ± 0.49**	90.88 ± 0.89	32.67 ± 1.13	**88.95**	34.64
ConfBranch [5]	92.85 ± 0.29	21.48 ± 0.94	68.90 ± 1.83	71.82 ± 3.39	90.43 ± 0.18	34.75 ± 0.63	84.06	42.68
G-ODIN [13]	**95.51 ± 0.31**	21.45 ± 1.91	**85.67 ± 1.58**	**42.68 ± 3.19**	**92.33 ± 0.11**	30.18 ± 0.49	**91.17**	**31.44**
LogitNorm [33]	**96.74 ± 0.06**	**13.81 ± 0.20**	81.53 ± 1.26	53.61 ± 3.45	**93.04 ± 0.21**	**26.11 ± 0.52**	**90.44**	**31.18**
CIDER [22]	**94.71 ± 0.36**	**20.72 ± 0.85**	80.49 ± 0.68	54.22 ± 1.24	90.66 ± 1.68	30.17 ± 2.75	88.62	35.04
Training methods with outliers								
OE [10]	**96.00 ± 0.13**	**13.13 ± 0.53**	81.41 ± 1.49	54.82 ± 2.79	89.02 ± 0.18	34.17 ± 0.56	88.81	**34.04**
MCD [36]	91.00 ± 1.10	32.03 ± 4.21	74.72 ± 0.78	54.39 ± 1.34	88.94 ± 0.10	29.93 ± 0.30	84.89	38.78
UDG [35]	**94.06 ± 0.90**	**20.35 ± 2.41**	79.59 ± 1.77	59.00 ± 3.35	82.09 ± 2.78	62.04 ± 5.99	85.25	47.13
MixOE [38]	91.93 ± 0.69	33.84 ± 4.77	76.40 ± 1.44	63.88 ± 2.48	88.27 ± 0.41	40.93 ± 0.29	85.53	46.22

As shown in Fig. 2(b), after convergence, the mean logits at ranks 0, 90, and 99 are approximately 53.72, −4.18, and −58.33, respectively, yielding a consistent minimal average rank separation margin of roughly 25 (difference of mean values between distributions). This is because ListMLE optimizes the pairwise logit ordering likelihood and therefore depends only on the relative margin between logits of adjacent ranks, penalizing margin violations more strongly than large positive margins [34]. Consequently, the optimization explicitly prioritizes preserving the correct ordering rather than pushing the correct class logit arbitrarily high. ListMLE progressively induces a structured semantic hierarchy along the rank dimension: logits associated with semantically closer classes remain comparatively high, while less similar classes are increasingly suppressed, leading to a stable and widening margin across the ranking spectrum. In contrast, low ranks (semantically nearest neighbors) in the CE model initially remain relatively higher, their separation decreases as training progresses, leading to substantial overlap in the logit distributions for mid- and high-rank classes and losing semantic ordering information. Moreover, as shown in Fig. 2(a), CE induces rank distribution compression, where semantically dissimilar classes become difficult to distinguish in later rank positions.



(a) CIFAR100 trained using standard cross-entropy CE loss - Logit distribution for different ranks



(b) CIFAR100 trained using ListMLE by progressive enforcement of ranking - logit distribution. (top): In distribution samples (bottom): Out of distribution samples.

Figure 2. Logit distributions at selected rank positions for predicted class 82 on CIFAR100. *Across training epochs, (b) ID samples show increasing separation across rank positions, while (b) OOD samples remain concentrated near zero with minimal separation.* OOD samples are shown for comparison only.

In the bottom row of Fig. 2(b), we report the rank-wise logit distributions for OOD samples. Unlike ID samples, OOD logits cluster near zero with substantially lower variance, forming a compact distribution. Nonetheless, a small yet consistent average margin between ranks remains. This is because ListMLE enforces relative logit margins only for ID samples, which indirectly shapes the geometry of the classifier weights. When an OOD feature lies closer to certain ID class regions, the corresponding logits reflect this proximity, producing slight rank-dependent variation. However, as shown in the Sec. 5.3 and Fig. 3, OOD samples do not maintain expected class ranking order for the predicted class.

Overall, the logit magnitude between the ID and OOD in different ranks enables effective OOD detection: ID samples maintain higher, stable rank-dependent margins, while OOD samples remain near zero across ranks.

5.3. Logit Ordering Consistency Explains Improved OOD Detection under RankOOD

The left column of Fig. 3 compares the distributions of MSP and RankOOD-S for CIFAR-10 (ID) and SVHN (OOD) samples under models trained with CE and RankOOD-T losses. Under CE training, MSP (**cf.** Fig. 3(a)) achieves an FPR95 of 24.24 on SVHN, while RankOOD reduces it by 37.12% (**cf.** Fig. 3(c)). The RankOOD-T loss explicitly enforces a consistent ranking structure among logits through ListMLE supervision. This encourages the logits to maintain smooth, monotonic decay rather than allowing a single confidence spike. Consequently, even when an OOD input produces a high top-1 logit, its intermediate logits tend to deviate from the canonical decay pattern observed in ID data. This structural deviation reduces the overall MSP score, shifting OOD MSP scores away from 1 and reducing FPR95 for MSP under RankOOD-T. Moreover, the RankOOD-S further decreases FPR95 to 12.23 (Fig. 3(e)). This occurs because the RankOOD-S amplifies this effect by penalizing rank-order violations and deviations from class-specific logit thresholds.

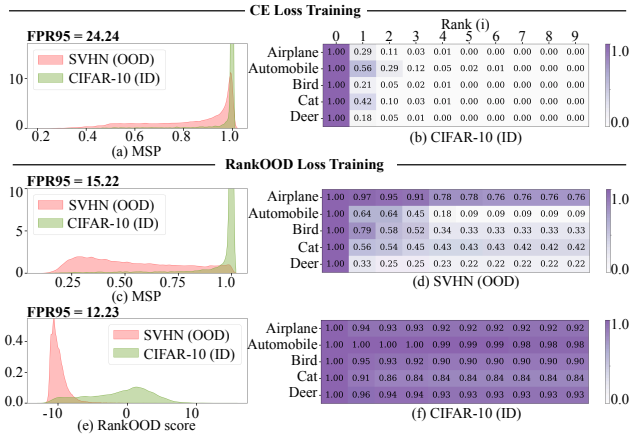


Figure 3. Left: Distributions of RankOOD-S and MSP scores for CIFAR-10 (ID) and SVHN (OOD) samples under RankOOD-T ad CE training. Right: Conditional probability matrix (CP) of rank position i given that all prior ranks have been correctly predicted.

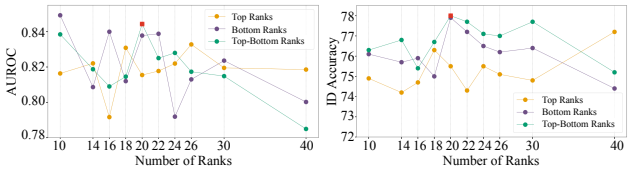


Figure 4. OOD and ID detection performance on CIFAR-100 when training with different rank subsets. (left): AUROC (right): ID accuracy. Top Ranks uses the top$-N$ class ranks, Bottom Ranks uses the lowest $N-1$ ranks along with rank-0, and Top-Bottom samples top-N/2 and lowest N/2 ranks.

To analyze how RankOOD leverages ranking consistency, the right column shows conditional probability (CP) matrices capturing $P(rank\ i \mid ranks < i\ are\ correct)$. We report three cases: (b) CIFAR-10 (ID) under CE training, (d) SVHN (OOD) under RankOOD-T, and (f) CIFAR-10 (ID) under RankOOD-T. Due to space constraints, we report CP matrices for five classes, and provide the complete figures in the Appendix A.6. Each CP matrix illustrates the model's ability to preserve the class-wise canonical rank order across ranks. While the CE-trained model (**cf.** Fig. 3(b)) shows weak rank-consistency, RankOOD-T induces sequential dependencies between ranks, yielding high conditional probabilities (> 0.84) for ID data (**cf.** Fig. 3(f)). In contrast, OOD samples under RankOOD-T loss exhibit early rank-order violations, accumulating higher penalties δ_{x_i}, in Eq. 5 and causing higher deviation from the reference logit thresholds Ref_i. These deviations contribute to RankOOD's superior sensitivity to OOD distortions.

5.4. Ablation Study

Subset of Ranks: In Fig. 4 (left), we evaluate OOD detection on CIFAR-100 when training with three types of rank subsets; Top ranks subset uses the top-N class ranks, Bottom ranks subset uses the lowest N-1 ranks with rank-0 and Top-Bottom combines the top-N/2 and botom-N/2 class ranks, respectively. Since the selection of rankings is not only crucial for OOD detection performance but also for ID accuracy, we illustrate the right sub-figure according to the same three subsets. As shown, using only the Top ranks yields lower ID accuracy since it focuses on separating semantically similar classes. In contrast, including Bottom or both Top–Bottom ranks improve ID accuracy by exposing the model to semantically dissimilar classes. Both the top and Bottom rank subsets exhibit unstable AUROC when trained with a small number of ranks. This behavior arises because the Top ranks primarily distinguish semantically similar class variations, leading to dominance of near-OOD samples, whereas the Bottom ranks, in high-level learns to differentiate highly dissimilar classes compared to rank-0 via logit separations, reducing sensitivity to near-OOD regions. When selecting Top-Bottom ranks, after a certain point, the performance decreases as the ranks

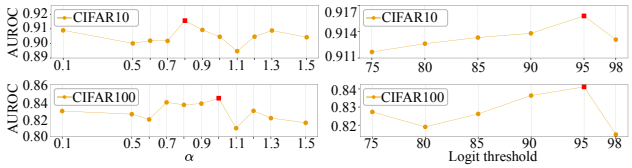


Figure 5. OOD detection performance on CIFAR-100 with respect to α and logit threshold (Ref) hyper-parameters.

introduce more noise. Empirically, selecting the top 10 and bottom 10 ranks (N=20) achieves the best balance - maximizing ID accuracy and maintaining high AUROC. This shows that enforcing Listwise ranking across semantically diverse ranks enhances OOD detection. We observed these numbers based on an $\alpha = 1.0$ and a logit threshold of 0.95.

Ablation of α and Logit threshold: In Fig. 5, we present the ablation results of selecting the alpha in Eq. 4 and the logit threshold (Ref) in the RankOOD-S. We conducted AUROC tests with different alpha and logit thresholds. For both datasets, $\alpha = 0.1$ shows high AUROC as the loss function is closer to CE loss, and the highest AUROC is achieved at 0.8 for CIFAR-10 and 1.0 for CIFAR-100. With both datasets, we observe AUROC increasing with the increments in the logit threshold, peaking at around 0.95 and starting to decline as the increasing Ref increases the gap between OOD rank logit and threshold in lower ranks, whereas ID samples exhibit a lower gap and vice versa for higher ranks. Consequently, the AUROC decreases once Ref surpasses this threshold, as the threshold becomes the maximum logit value of the given rank. This variation demonstrates significant performance improvement.

Effectiveness of RankOOD-T: We evaluate RankOOD-T with various logit-based OOD scores and consistently outperform standard CE training. These results suggest that primary gains are due to the RankOOD-T objective ; detailed results are provided in Appendix A.3.

6. Concluding Remarks

In this paper, we presented a novel OOD detection framework, *RankOOD*, which leverages class-wise logit ranking structures to enhance the detection of out-of-distribution samples. RankOOD constructs canonical rank orders using a 0–1 integer linear programming formulation guided by semantic similarity from probability mass functions of a pretrained network, and optimizes them via a Listwise Maximum Likelihood Estimation (ListMLE) objective. Extensive evaluations on CIFAR-10, CIFAR-100, and TinyImageNet demonstrate that RankOOD consistently ranks among the top two methods for near-OOD detection and the top three for far-OOD detection across 34 baselines. Notably, RankOOD achieves superior AUROC and FPR95 on the challenging TinyImageNet near-OOD benchmark, highlighting its ability to capture fine-grained semantic distinctions between ID and OOD samples.

References

[1] Abhijit Bendale et al. Towards open set deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1563–1572, 2016. 1

[2] Julian Bitterwolf, Maximilian Müller, and Matthias Hein. In or out? fixing imagenet out-of-distribution detection evaluation. In *Proceedings of the 40th International Conference on Machine Learning*. JMLR.org, 2023. 5

[3] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, 2014. 5

[4] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 5

[5] Terrance DeVries et al. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018. 5, 6

[6] Andrija Djurisic, Nebojsa Bozanic, Arjun Ashok, and Rosanne Liu. Extremely simple activation shaping for out-of-distribution detection. In *The Eleventh International Conference on Learning Representations*, 2023. 2, 5, 6

[7] 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 - Volume 70*, page 1321–1330. JMLR.org, 2017. 1, 6

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 5

[9] 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. 1, 2, 5, 6

[10] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2019. 1, 2, 5, 6

[11] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *Advances in neural information processing systems*, 32, 2019. 2

[12] Dan Hendrycks, Steven Basart, Mantas Mazeika, Andy Zou, Joseph Kwon, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. Scaling out-of-distribution detection for real-world settings. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, pages 8759–8773. PMLR, 2022. 6

[13] Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10951–10960, 2020. 2, 5, 6

[14] Naveen Karunanayake, Suranga Seneviratne, and Sanjay Chawla. Excel: Combined extreme and collective logit information for enhancing out-of-distribution detection. *arXiv preprint arXiv:2311.14754*, 2023. 1, 3, 5, 6

[15] Naveen Karunanayake, Suranga Seneviratne, and Sanjay Chawla. Craft: Class ranking aware fine-tuning for enhanced out-of-distribution detection. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4119–4128. IEEE, 2025. 1, 2, 3, 5, 6

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 4

[17] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 4

[18] 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. 2

[19] Weitang Liu, Xiaoyun Wang, John D. Owens, and Yixuan Li. Energy-based out-of-distribution detection. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2020. 1, 2, 5, 6

[20] Xixi Liu et al. Gen: Pushing the limits of softmax-based out-of-distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23946–23955, 2023. 5, 6

[21] John I Marden. *Analyzing and modeling rank data*. CRC Press, 1996. 2, 3, 4

[22] Yifei Ming, Yiyou Sun, Ousmane Dia, and Yixuan Li. How to exploit hyperspherical embeddings for out-of-distribution detection? In *The Eleventh International Conference on Learning Representations*, 2023. 2, 6

[23] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. 5

[24] Jie Ren, Stanislav Fort, Jeremiah Liu, Abhijit Guha Roy, Shreyas Padhy, and Balaji Lakshminarayanan. A simple fix to mahalanobis distance for improving near-ood detection. *arXiv preprint arXiv:2106.09022*, 2021. 6

[25] Youssef Shoeb, Azarm Nowzad, and Hanno Gottschalk. Out-of-distribution segmentation in autonomous driving: Problems and state of the art. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 4310–4320, 2025. 1

[26] Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. In *International Conference on Machine Learning*, pages 20827–20840. PMLR, 2022. 6

[27] Yiyou Sun et al. React: Out-of-distribution detection with rectified activations. *Advances in Neural Information Processing Systems*, 34:144–157, 2021. 1, 2, 5, 6

[28] Yiyou Sun et al. Dice: Leveraging sparsification for out-of-distribution detection. In *European Conference on Computer Vision*, pages 691–708. Springer, 2022. 2

[29] Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. Csi: Novelty detection via contrastive learning on distributionally shifted instances. *Advances in Neural Information Processing Systems*, 33:11839–11852, 2020. 2

[30] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and

Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8769–8778, 2018. 5

[31] Sagar Vaze, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Open-set recognition: A good closed-set classifier is all you need. In *International Conference on Learning Representations*, 2022. 5

[32] Haoqi Wang, Zhizhong Li, Litong Feng, and Wayne Zhang. Vim: Out-of-distribution with virtual-logit matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4921–4930, 2022. 1, 5, 6

[33] Hongxin Wei, Renchunzi Xie, Hao Cheng, Lei Feng, Bo An, and Yixuan Li. Mitigating neural network overconfidence with logit normalization. In *International Conference on Machine Learning*, pages 23631–23644. PMLR, 2022. 2, 5, 6

[34] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199, 2008. 2, 3, 6

[35] Jingkang Yang, Haoqi Wang, Litong Feng, Xiaopeng Yan, Huabin Zheng, Wayne Zhang, and Ziwei Liu. Semantically coherent out-of-distribution detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8301–8309, 2021. 1, 2, 5, 6

[36] Qing Yu et al. Unsupervised out-of-distribution detection by maximum classifier discrepancy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9518–9526, 2019. 1, 2, 5, 6

[37] Lifan Yuan, Yangyi Chen, Ganqu Cui, Hongcheng Gao, Fangyuan Zou, Xingyi Cheng, Heng Ji, Zhiyuan Liu, and Maosong Sun. Revisiting out-of-distribution robustness in nlp: Benchmarks, analysis, and llms evaluations. *Advances in Neural Information Processing Systems*, 36:58478–58507, 2023. 1

[38] Jingyang Zhang, Nathan Inkawhich, Randolph Linderman, Yiran Chen, and Hai Li. Mixture outlier exposure: Towards out-of-distribution detection in fine-grained environments. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5531–5540, 2023. 1, 2, 5, 6

[39] Jingyang Zhang, Jingkang Yang, Pengyun Wang, Haoqi Wang, Yueqian Lin, Haoran Zhang, Yiyou Sun, Xuefeng Du, Kaiyang Zhou, Wayne Zhang, et al. Openood v1. 5: Enhanced benchmark for out-of-distribution detection. *arXiv preprint arXiv:2306.09301*, 2023. 2, 4

[40] 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*, 40(6):1452–1464, 2017. 5

Acknowledgment

This research was supported by the Australian Research Council's Discovery Project's funding scheme (Project ID DP220102520).