Data Specified Feature Rectification for Out-of-distribution Detection

Anonymous CVPR submission

Paper ID 2062

Abstract

Out-of-distribution (OOD) detection is an anomaly-handling mechanism, where classification systems should detect outliers with true labels outside the label space, distinguishing them from normal in-distribution (ID) samples. Advanced works have observed that, even with fixed model parameters, one can achieve compelling detection performance by rectifying misleading features. *Typically, such misleading features are identified and corrected via pruning a portion of features or setting their maximal values. However, such restricted forms of clamping may limit the true power of features rectification, hindering subsequent studies. To this end, we propose* Instance-customized Rectification *(InsRect), suggesting a more general structure that can tailor rectification for varying instances. Specifically, InsRect focuses on common components, of which distinct combinations well reconstruct instance features of our interest (e.g., principal components). Then, we specify minimal/maximal-allowed values for respective common components, thereby empowering instance customization due to reconstruction (via varying combinations) after clamping (for common components). Overall, InsRect leads to a flexible framework of features rectification without requiring excessive number of parameters, thus simple yet effective in practice. We conduct extensive experiments for various OOD detection setups on CIFAR and ImageNet-1K benchmarks, demonstrating the power of InsRect against state-of-the-art counterparts.*

1. Introduction

Deep learning systems learn from in-distribution (ID) data during training but inevitably encounter out-of-distribution (OOD) data when deployed. It leads to the OOD detection task [11, 35], where the systems should identify OOD data as anomalies meanwhile make accurate label predictions for ID data [2]. Nowadays, OOD detection has attracted intensive attention due to its crucial role for many real-world applications that involve safety and reliability, such as autonomous driving [10] and medical analysis [63].

OOD detection remains challenging due to the well-known calibration failures [12, 14]—deep models can assign arbitrary-high confidence to OOD data [1, 37], making labeling confidence unreliable in OOD detection. Therefore, a lot of efforts have been paid to overcome these obstacles and existing works can generally be attributed into two categories, namely, *fine-tuning* approaches and *post-hoc* approaches. Fine-tuning approaches study various model training methods to boost OOD detection [3, 10, 15, 26, 32, 43, 52, 57], but incurring additional costs of training and re-deployment. By contrast, post-hoc approaches directly use model outputs from a fixed model, designing scoring functions on top of models to indicate OOD [8, 14, 27, 30, 31, 47, 49, 55]. Post-hoc approaches are generally easy for deployment yet effective in OOD detection, thus are the main focus of this paper.

Features rectification [8, 49] in post-hoc OOD detection attracts emerging attentions nowadays. The key observation is that model features contain misleading information during forward propagation, confusing scoring functions with deficient detection performance. Therefore, features rectification focuses on correcting such misleading information in a post-hoc manner. Overall, existing works [8, 49] conduct features rectification in the penultimate layer, assuming that extreme features elements (with very large or small values) are confusing and should be clamped by the predefined thresholds, cf., Figure 1 (ReAct [49] and ASH [8]). These methods have achieced many promising results in post-hoc OOD detection, especially when facing large ID semantic space [16, 50]. However, their forms of features rectifications are relatively limited, potentially indicating a large space of further improvement. It motivates us to ask the following question:

How to generalize previous works in features rectification with more flexible forms and better performance?

To this end, we propose *Instance-customized Rectification* (InsRect), a novel features rectification strategy having controllable number of parameters yet remaining flexible. In general, we introduce a set of *common components*, whose different weighted combinations can well

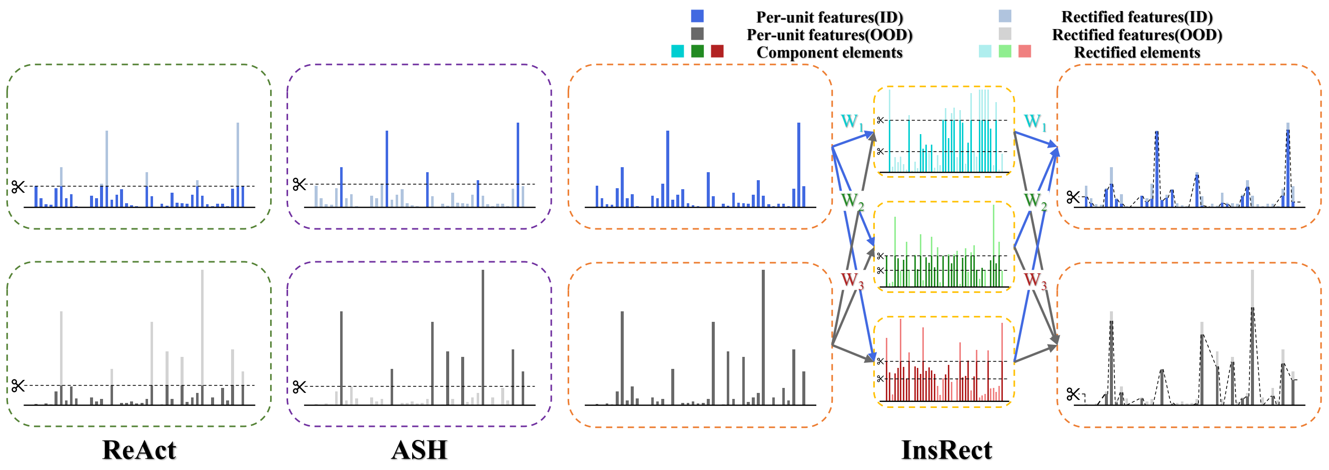


Figure 1. **Comparison between ReAct, ASH and our InsRect**. We depict the embedding features for two data points (blue and grey respectively) and illustrate the impacts of features rectification for ReAct, ASH, and our InsRect. As we can see, ReAct and ASH both propose clamping the plain features directly with upper/bottom thresholds. Our InsRect goes one step further, clamping common components instead for a more general form of features rectification. Since distinct weighted combinations (i.e., w_1, w_2, w_3) are associated with different data points, rectified common components have different impacts on them, thus empowering instance-customized features rectification.

approximate original features, e.g., via principal component analysis (PCA) [59], non-negative matrix factorization (NMF) [28] or independent component analysis (ICA) [44]. Then, as in Figure 1, instead of directly clamping plain features as in previous works, we specify minimal/maximal-allowed values for each common component instead. The number of additional parameters in InsRect only depends on the number of common components, thus with controllable number of parameters to be tuned. Overall, we can define a set of common components requiring to be rectified and assign specific rectification strategies on them, which is more general than previous works in rectifying plain features. Moreover, clamping common components has different impacts of rectification for different instances, since different instances correspond to different weighted combinations of common components. Thus, InsRect also empowers instance-customized features rectification (as our name suggested), which is largely overlooked in previous works.

InsRect is powerful even when the number of common components is small (e.g., $1 \sim 5$), but we are also interested in the cases where we have many more common components (e.g., $10 \sim 100$). However, manual tuning can be hard since the number of parameters grows linearly w.r.t. the number of common components. Therefore, we suggest Bayesian optimization (BO) [46] for choosing InsRect parameters automatically, which is superior to gradient-based optimization since gradient calculation is generally intractable for thresholding. BO enables a simple evaluation of InsRect performance with extreme number of common components, helping us to understand its impacts on OOD detection and motivating the following works.

We conduct extensive experiments in Section 4 on widely used benchmark datasets, verifying our effectiveness against a wide range of different post-hoc approaches for OOD detection. For common OOD detection, our InsRect reduces the average FPR95 by 4.60, 13.46, and 3.85 compared with the current best scoring functions on CIFAR-10, CIFAR-100, and ImageNet-1K. For hard OOD detection, our InsRect reduces the average FPR95 by 2.74 and 11.14 compared with the current best scoring functions on CIFAR-10 and CIFAR-100, respectively.

2. Preliminaries

Let $\mathcal{X} \subseteq \mathbb{R}^d$ the input space and $\mathcal{Y} = \{1, 2, ..., C\}$ the label space. We assume the ID joint distribution $\mathcal{P}^{\text{ID}}_{\mathcal{X},\mathcal{Y}}$ defined over $\mathcal{X} \times \mathcal{Y}$ and the OOD joint distribution $\mathcal{P}^{\text{OOD}}_{\mathcal{X},\mathcal{Y}'}$ defined over $\mathcal{X} \times \mathcal{Y}'$ where $\mathcal{Y} \cap \mathcal{Y}' = \emptyset$. We also have a deep model $\mathbf{f}$ (logit outputs) well-trained in a close-world setup, i.e., trained over $\mathcal{P}^{\text{ID}}_{\mathcal{X},\mathcal{Y}}$ for the label classification across $\mathcal{Y}$. Moreover, we assume the model can be represented by $\mathbf{f} = \mathbf{h} \circ \mathbf{e}$ with $\mathbf{e}$ the features extractor and $\mathbf{h}$ the classifier.

In general, the model $\mathbf{f}$ can make predictions for most data from $\mathcal{P}^{\text{ID}}_{\mathcal{X}}$ yet fail for classification given any data from $\mathcal{P}^{\text{OOD}}_{\mathcal{X}}$. However, when the model $\mathbf{f}$ is deployed, it may encounter a mixture of the distribution $\mathcal{P}^{\text{ID}}_{\mathcal{X}}$ and the distribution $\mathcal{P}^{\text{OOD}}_{\mathcal{X}}$. Hence, the close-world model $\mathbf{f}$ may encounter OOD data where itself cannot handle in the open world.

2.1. Scoring Functions

OOD detection hopes that the close-world model $\mathbf{f}$ will remain reliable when deployed in the open world. It generally expects that the classification system should not only make accurate label predictions for ID data meanwhile detect OOD data without making further predictions. Overall, given the well-trained $\mathbf{f}$ on ID data, we typically design various scoring functions towards effective OOD detection. Following [31], OOD data should have higher scores than ID data, and here we present several promising examples.

Maximum Softmax Prediction (MSP) takes the maximal value of $\mathrm{softmax}$ outputs to indicate the ID confidence:

$$s_{\texttt{MSP}}(\mathbf{x};\mathbf{f}) = -\max_i \mathrm{softmax}_i\,\mathbf{f}(\mathbf{x}), \tag{1}$$

where $\mathrm{softmax}_i(\cdot)$ denotes the i-th element of the $\mathrm{softmax}$ outputs. Ideally, ID data have true labels in the considered label space, thus should have high confident predictions and low MSP scores. However, due to calibration failures [12], previous works have found that MSP is unreliable, motivating subsequent works that overcome its drawbacks.

Free Energy Scoring (Energy) takes the free energy function for OOD scoring, which processes the logit outputs with the log-sum-exp operation, following

$$s_{\texttt{Energy}}(\mathbf{x};\mathbf{f}) = -\log\sum_i \exp f_i(\mathbf{x}), \tag{2}$$

where $f_i(\cdot)$ denotes the i-th element of the output logits. Compared to MSP, the energy score better covers the covariate distribution of ID data, thus less susceptible to the over-confidence issue inherited in MSP.

Rectified Activation (ReAct) further assumes the existence of misleading information in model outputs, which should be rectified by truncating their large elements, i.e.,

$$s_{\texttt{ReAct}}(\mathbf{x},c;\mathbf{f}) = -\log\sum_i \exp h_i\big(\min(\mathbf{e}(\mathbf{x}),c)\big), \tag{3}$$

where $\max(\cdot,c)$ is a clamping operator that ensures the maximal element of $\mathbf{e}(\mathbf{x})$ should be smaller than c. Since the misleading information is inhibited, ReAct can largely improve free energy scoring, especially for many challenging OOD detection tasks.

Activation Shaping (ASH) also assumes the misleading information in model outputs but prunes a portion of small features instead, which is given by

$$e_{\texttt{ASH},i}(\mathbf{x},p) = \begin{cases} e_i(\mathbf{x}) & e_i(\mathbf{x}) \geq \rho\big(\mathbf{e}(\mathbf{x}),p\big) \\ 0 & e_i(\mathbf{x}) < \rho\big(\mathbf{e}(\mathbf{x}),p\big) \end{cases} \tag{4}$$

where $\rho(\cdot,p)$ is an operator that selects the p-th percentile of inputs and $e_i(\mathbf{x})$ refers to the i-th element of $\mathbf{e}(\mathbf{x})$. Thereafter, ASH also adopts the free energy scoring in OOD detection, i.e.,

$$s_{\texttt{ASH}}(\mathbf{x},p;\mathbf{f}) = -\log\sum_i \exp h_i\big(\mathbf{e}_{\texttt{ASH}}(\mathbf{x},p)\big). \tag{5}$$

ReAct and ASH both focus on features rectification, truncating or pruning extreme elements for the embedding outputs. Although simple, these advanced works have achieved promising performance across varying OOD detection tasks. However, their restricted forms of truncating or pruning may limit the true power of features rectification, potentially indicating a large space for their further improvements. It motivates us to study how to generalize previous works, devising more flexible and effective framework of features rectification.

3. Our Method

To this end, we propose *Instance-customized Rectification strategy* (InsRect), which is a novel features rectification scheme for post-hoc OOD detection. InsRect is more flexible than previous works by clamping a pre-defined set of *common components* in the embedding space that are shared across ID data, thereafter recovering to the original features space after clamping to construct rectified features (c.f., Figure 1). Overall, the superiority of our method is supported by two factors in the following.

- Rectifying misleading information in common components are more flexible than correcting original features, since features components typically contribute to semantic/conceptual understanding instead of individual features elements [51].
- Clamping common components leads to instance customized features rectification without introducing much computing overhead. Embedding features are of different weight combination of common components, and thus rectifying common components have different impacts for varying instances [60].

We discuss our InsRect below, talking about common components, features rectification, and the tuning strategy.

3.1. Rectification Scheme

Clamping a set of common components instead of original embedding features is the key to our method. Therefore, we introduce the concepts of common components first.

Common Components. Following previous works [8], the embedding features space $\mathbb{R}^q$ is the basis for features rectification, defined by the penultimate layer, i.e., $\mathbf{e}(\mathbf{x})$, after its ReLU activation layer. As aforementioned, we consider a set of common components of size J, i.e., $\{\mathbf{c}_j\}_{j=1}^J$ in $\mathbb{R}^q$, of which linear combinations can well reconstruct original embedding features for ID data. Formally speaking, considering a set of ID embedding features of the size K, i.e., $\{\mathbf{e}(\mathbf{x}_k)\}_{k=1}^K$, common components should enjoy a sufficiently small reconstruction error, i.e.,

$$\min_{\{\mathbf{c}_j\}_j,\{\mathbf{w}_k\}_k} \sum_k \zeta\big(\mathbf{e}(\mathbf{x}_k), \sum_j w_{k,j}\mathbf{c}_j\big) \tag{6}$$

is small, where $\zeta(\cdot,\cdot)$ is a distance metric in the embedding features space $\mathbb{R}^q$ and $\mathbf{w}_k$ denotes the weighting strategy w.r.t. $\{\mathbf{c}_j\}_j$ to reconstruct $\mathbf{x}_k$. Overall, common components should be the most principal parts that represent ID data in the embedding space, thus containing critical information in OOD detection. Please refer to Section 3.2 for the detailed realization to compute common components. However, common components may still subject to misleading information, where we should impose features rectification therein to facilitate OOD detection.

Component Rectification. Now, we discuss the rectification method for common components. Based on previous works [8, 49], we also assume extreme values attribute to misleading information for OOD detection, but with respect to common components. Therefore, for each common component $\mathbf{c}_j$, we assign the bottom thresholds a_j and the upper thresholds r_j, following the rectification strategy in each dimension i of the form

$$\bar{c}_{j,i} = \begin{cases} r_j & c_{j,i} \geq r_j \\ c_{j,i} & a_j \leq c_{j,i} < r_j \\ 0 & c_{j,i} < a_j \end{cases}, \tag{7}$$

where $c_{j,i}$ denotes the i-th element before rectification and $\bar{c}_{j,i}$ denotes that after rectification. Note that $\{a_j\}_j$ and $\{r_j\}_j$ are parameters that can be tuned manually when J is small. Moreover, in Section 3.3, we further discuss the automated tuning strategy for $\{a_j\}_j$ and $\{r_j\}_j$, which is especially helpful when J is large relatively.

Embedding Rectification. The rectified common components further facilitate rectification of embedding features, where we discern two parts for the embedding features $\mathbf{e}(\mathbf{x}_k)$: the *reconstructing part* $\sum_j w_{k,j}\mathbf{c}_j$ that recovers original features by common components and the *error part* $\boldsymbol{\epsilon}_k = \mathbf{e}(\mathbf{x}_k) - \sum_j w_{k,j}\mathbf{c}_j$ that models the bias after reconstruction. Since features rectification is considered only for common components, we conduct embedding rectification for the reconstructing part, substituting $\{\mathbf{c}_j\}_j$ by rectified counterparts of $\{\bar{\mathbf{c}}_j\}_j$. Therefore, the rectified embedding features are

$$\mathbf{e}_{\texttt{InsRect}}(\mathbf{x}_k) = \boldsymbol{\epsilon}_k + \sum_j w_{k,j}\bar{\mathbf{c}}_j, \tag{8}$$

for the original embedding features $\mathbf{e}(\mathbf{x}_k)$. Overall, Eq. (8) benefits from the rectification of common components, which can facilitate OOD detection as aforementioned.

InsRect Scoring. After our rectification scheme, the embedding features can be used to boost many conventionally used scoring strategy. Following previous works [8], the free energy scoring is used as the basis by default, i.e.,

$$s_{\texttt{InsRect}}(\mathbf{x}_k) = -\log\sum_i \exp h_i\big(\mathbf{e}_{\texttt{InsRect}}(\mathbf{x}_k)\big), \tag{9}$$

which facilitates OOD detection with rectified embedding features. Moreover, we also find that the scaling operation, first introduced in [8], can further facilitate our InsRect score, used for the embedding features $\mathbf{e}_{\texttt{InsRect}}(\mathbf{x}_k)$ in our realization. Please refer to Appendix G for detailed discussion about features scaling.

3.2. Common Components and Instance Weights

The suggested features rectification scheme is general, depending on different methods in finding common components. In this paper, we take NMF as an example, which is suitable for our InsRect scoring since the embedding features after ReLU activation layer are all non-negative. Moreover, compared with PCA and ICA, NMF can better capture the additive nature of data, discovering part-based representation for critical features extraction. Please refer to Section 4.3 for comparing PCA, NMF and ICA in constructing common components.

Computing $\{\mathbf{c}_j\}_j$ for training data. NMF can be viewed as a particular realization of Eq. 6 with ζ the Euclidean distance, which can be written as

$$\min_{\{\mathbf{c}_j\}_j,\{\mathbf{w}_k\}_k} \sum_k ||\mathbf{e}(\mathbf{x}_k) - \sum_j w_{k,j}\mathbf{c}_j||. \tag{10}$$

Such an optimization problem is typically solved in a bi-level manner, optimizing $\{\mathbf{c}_j\}_j$ and $\{\mathbf{w}_k\}_k$ alternatively. Specifically, in the t-th step, common components and the instance-associated weights are respectively updated via

$$c_{j,i}^{(t+1)} \leftarrow c_{j,i}^{(t)} \frac{\mathbf{w}_{\cdot,j}^{(t)}\left[\mathbf{e}_j\left(\mathbf{x}_k\right)\right]_j}{\mathbf{w}^{(t)}{}_{\cdot,j}(\sum_j \mathbf{w}^{(t)}{}_{\cdot,j}c_{j,i}^{(t)})} \tag{11}$$

and

$$w_{k,j}^{(t+1)} \leftarrow w_{k,j}^{(t)} \frac{\mathbf{e}(\mathbf{x}_k)\mathbf{c}_j^{(t+1)}}{(\sum_j w_{k,j}^{(t)}\mathbf{c}_j^{(t+1)})\mathbf{c}_j^{(t+1)}}, \tag{12}$$

We iterate between Eq. 11 and Eq. 12 until convergence or reach the maximal iteration steps T.

Computing w for test data. We preserve only $\{\mathbf{c}_j\}_j$ after optimization, which are taken as common components. Then, given a new test data, we need to compute associated weights for embedding rectification mentioned in Section 3.1. However, the optimal $\mathbf{w}$ in NMF has no closed-form solution, which should be taken as an optimization problem following Eq. 10 with fixed $\{\mathbf{c}_j\}_j$. Specifically, for a new instance $\mathbf{x}_k$, the optimal weight $\mathbf{w}_k$ is computed by solving

$$\min_{\mathbf{w}_k} ||\mathbf{e}(\mathbf{x}_k) - \sum_j \mathbf{w}_{k,j}\mathbf{c}_j|| \tag{13}$$

which can be solved iteratively, following,

$$w_{k,j}^{(t+1)} \leftarrow w_{k,j}^{(t)} \frac{\mathbf{e}(\mathbf{x}_k)\mathbf{c}_j}{(\sum_j w_{k,j}^{(t)}\mathbf{c}_j)\mathbf{c}_j}, \tag{14}$$

Algorithm 1 Instance-customized Rectification Scoring

```
Require: Common components $\{\mathbf{c}_j\}_j$, clamping thresholds $\{a_j\}_j, \{r_j\}_j$, test input $\mathbf{x}_k$.
1: Compute rectified components $\{\bar{c}_j\}_j$ in Eq. (7);
2: Compute embedding features $\mathbf{e} = \mathbf{e}(\mathbf{x}_k)$;
3: for $t = 1 \dots T$ do
4:     Update $w_{k,j}^{(t+1)} \leftarrow w_{k,j}^{(t)} \frac{\mathbf{e}(\mathbf{x}_k)\mathbf{c}_j}{(\sum_j w_{k,j}^{(t)}\mathbf{c}_j)\mathbf{c}_j}$;
5: end for
6: Compute error part via $\boldsymbol{\epsilon} = \mathbf{e} - \sum_j w_{k,j}^{(T)}\mathbf{c}_j$
7: Compute rectified embedding via $\bar{\mathbf{e}} = \boldsymbol{\epsilon} + \sum_j w_{k,j}^{(T)}\bar{\mathbf{c}}_j$
8: Compute rectified logits via $\bar{\mathbf{f}} = \mathbf{h}(\bar{\mathbf{e}})$.
9: return $-\log \sum_i \exp \bar{f}_i$.
```

until convergence as in Eq. (13).

Overall Algorithm. We summarize the InsRect score computation in Algorithm 1, mainly consisting of 5 steps as follows: in **Step 1**, we rectify common components with the given clamping thresholds; in **Step 4**, we iterate the updating rule to find the proper weights; in **Step 6**, we compute the error part that does not need rectification; in **Step 7**, we reconstruct the embedding features after rectification; in **Step 8**, we compute the resulting logit outputs, and then return the InsRect score.

3.3. Bayesian Optimization

The clamping thresholds $\{a_j\}_j$ and $\{r_j\}_j$ can be tuned manually when the number of common components, i.e., J, is small. However, large values of J are also of interest to explore the full capability of InsRect. In this case, manual tuning is prohibitive, and an automatic tuning strategy should be given to reduce human labors.

In our work, we adopt BO for thresholds searching, finding the proper values for $\{a_j\}_j$ and $\{r_j\}_j$ from their predefined searching ranges. It is superior to gradient-based methods since the gradient computation of thresholding is intractable. Following the conventional setup of outlier exposure [15], we adopt the FPR95 computed with ID training data (i.e., $\{\mathbf{x}_k^{\mathrm{I}}\}_k$) and auxiliary OOD data (i.e., $\{\mathbf{x}_k^{\mathrm{O}}\}_k$) to optimize thresholds. The optimization objective is given by

$$\frac{\sum_k \mathbf{I}[s_{\texttt{InsRect}}(\mathbf{x}_k^{\mathrm{O}}) < \tau]}{\sum_k \mathbf{I}[s_{\texttt{InsRect}}(\mathbf{x}_k^{\mathrm{O}}) < \tau] + \sum_k \mathbf{I}[s_{\texttt{InsRect}}(\mathbf{x}_k^{\mathrm{I}}) \geq \tau]}, \tag{15}$$

where $\mathbf{I}$ is the indicator function and τ is a threshold chosen such that the TPR, i.e.,

$$\frac{\sum_k \mathbf{I}[s_{\texttt{InsRect}}(\mathbf{x}_k^{\mathrm{O}}) \geq \tau]}{\sum_k \mathbf{I}[s_{\texttt{InsRect}}(\mathbf{x}_k^{\mathrm{O}}) \geq \tau] + \sum_k \mathbf{I}[s_{\texttt{InsRect}}(\mathbf{x}_k^{\mathrm{I}}) < \tau]}, \tag{16}$$

is near 0.95. The optimization objective is to minimize Eq. (15) in discerning ID and OOD data, i.e., large InsRect scores for OOD and small InsRect scores for ID. Please refer to Appendix E for the detailed discussion for BO.

4. Experiment

This section conducts extensive experiments to evaluate InsRect. In Section 4.1, we describe the basic experimental setups. In Section 4.2, we compare our method against representative counterparts on both CIFAR [24] and ImageNet-1K [7] benchmarks. In Section 4.3, we further conduct ablation studies for a comprehensive analysis of InsRect.

4.1. Setups

To begin with, we present the evaluation setups for InsRect.

Baselines. We compare with representative post-hoc methods in OOD detection, including MSP [14], MaxLogit [16], ODIN [30], Mahalanobis [27], Energy [31], GradNorm [21], ReAct [49], DICE [48], and ASH [8].

Datasets and Models. We mainly follow the detection setups given by [8]: on the CIFAR benchmark, we employ Textures [4], Places365 [67], LSUN-Crop [64], LSUN-Resize [64], iSUN [62], and SVHN [36] as test OOD datasets. By default, we adopt DenseNet-101 [19] with pre-trained parameters given by DICE repository as the backbone. On ImageNet-1K benchmark, we employ Textures [4], Places365 [67], iNaturalist [18], and SUN [61] as test OOD datasets. We adopt ResNet-50 [13] and MobileNetV2 [41] with pretrained parameters provided by Pytorch[1] as the backbones, if no otherwise specified. Moreover, for the auxiliary OOD datasets, which are used for BO search in Section 3.3, we follow previous works [45, 57] in using the tiny-ImageNet dataset [25] for the CIFAR benchmark and the ImageNet-21K-P dataset [40] for the ImageNet-1K benchmark.

Hyper-parameter Setups. By default, we use BO for hyper-parameter search, which is superior to manual tuning with wider candidates of J. For more results with manual tuning, please refer to Appendix J. Moreover, to ease the search procedure, each a_j (or r_j) is given by an associated proportion p_{a_j} (or p_{r_j}), such that $a_j = \rho(\{\mathbf{c}_j\}_{j=1}^J, p_{a_j})$ (or $r_j = \rho(\{\mathbf{c}_j\}_{j=1}^J, p_{r_j})$). Then, we define two search ranges as $p_{a_j} \in (0, u - 10^{-3})$ and $p_{r_j} \in (u + 10^{-3}, 100)$, fixing $u = 65$. Moreover, J is 10 for CIFAR-10 and 50 for both CIFAR-100 and ImageNet-1K setups; T is fixed to 500 across different considered setups. Note that no need to make fine-grained search since our InsRect is pretty robust to varying hyper-parameters (cf., Section 4.3).

Evaluation Metrics. As a common practice, the performance of OOD detection is evaluated with two representative metrics, i.e., FPR95 and AUROC, which are both threshold-independent [6]. FPR95 is the false positive rate when the true positive rate is at 95%; AUROC is the area

[1]`https://pytorch.org`

Table 1. Comparison with representative OOD detection methods on both the CIFAR and the ImageNet-1K benchmarks. ↓ (or ↑) indicates smaller (or larger) values are preferred. The best result in each column is indicated in bold.

Methods	CIFAR-10		CIFAR-100		ImageNet-1K (ResNet-50)		ImageNet-1K (MobileNetV2)	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
MSP	23.84	91.96	75.61	67.63	56.99	82.75	60.64	80.64
MaxLogit	21.09	94.51	68.04	75.06	48.06	87.00	47.98	86.37
ODIN	20.81	94.74	66.58	76.30	47.56	87.73	47.54	87.01
Mahalanobis	58.07	80.13	58.44	78.35	65.19	70.77	94.54	42.69
Energy	20.98	94.52	67.88	75.22	48.31	86.97	47.91	86.56
GradNorm	21.13	94.44	67.02	75.57	62.26	78.03	60.91	77.37
ReAct	21.55	94.16	51.63	81.57	30.63	93.29	43.54	88.74
DICE	21.19	95.49	53.56	85.47	62.66	84.97	52.86	87.87
ASH	16.58	95.85	40.98	87.59	23.91	94.86	37.62	90.20
InsRect	**11.98**	**97.36**	**27.52**	**92.88**	**20.06**	**96.04**	**34.57**	**91.55**

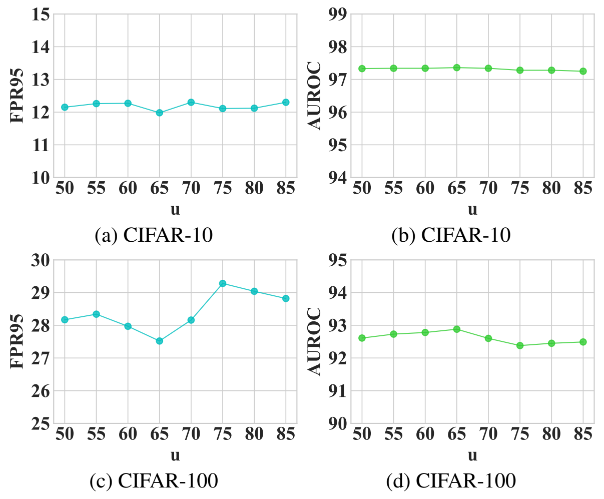


Figure 2. The performance of OOD detection with different hyperparameter u varying from 50 to 85. As we can see, the performance remains relatively steady over a large range, which implies that the method for finding clamping thresholds is stable.

under the receiver operating characteristic curve. Please refer to [17] for their more discussions.

4.2. Main Results

Now, we present our main comparison between InsRect and representative works in OOD detection.

Common OOD detection. We summarize the main results on both the CIFAR and the ImageNet-1K benchmarks in Table 1. Among the baseline methods, we observe that features rectification, i.e., ReAct and ASH, can consistently achieve better or comparable performance than all other previous works. Moreover, comparing between InsRect and these previous methods in features rectification, we further observe that InsRect achieves at least 4.60, 13.46 and 3.85 improvements on CIFAR-10, CIFAR-100 and ImageNet-1K (ResNet-50), measured by FPR95. It demonstrates the effectiveness of InsRect in features rectification.

Hard OOD detection. Besides the common OOD datasets for evaluations, we also consider hard OOD scenarios [52], of which the OOD datasets for testing are relatively similar to ID datasets in semantics or styles. Specifically, we adopt CIFAR-10 and CIFAR-100 as ID datasets; ImageNet-Fix [7], ImageNet-Resize [7], and Oxford-Pets [38] as test OOD datasets. As we can see from Table 2 and Table 3, InsRect can still outperform the considered baselines, which demonstrates the reliability of InsRect in OOD detection.

Other backbones. To demonstrate the general effectiveness of our InsRect, we also conduct experiments across varying backbones on CIFAR-100 and ImageNet-1K. Please refer to Appendix D for more results on CIFAR-10 and refer to Appendix C for pre-training setups of additional backbones. As we can see, on CIFAR-100, we adopt ResNet-50 [13], WRN-40-2 [66]; on ImageNet-1K, we adopt DenseNet-161 [19], WRN-50-2 [66]. We demonstrate the average results on test OOD datasets in Tables 4-5, where we find that our InsRect can still outperform the baselines across all these setups. It reveals that our InsRect is stable across varying backbones, further verifying our superiority.

4.3. Ablation Studies

we further present parametric stability analysis and ablation studies, trying to have a better understanding of our method. Note that, the OOD performance provided below is evaluated by the test ID datasets and a part of OOD data separated from the associated test OOD datasets.

Number of Common Components. We assume the fixed J throughout our previous experiments, while the effects for varying numbers of common components are also of our interest. As shown in Table 7, larger J generally leads to better OOD performance, while the performance gain shrinks thereby. Moreover, larger J also indicates larger

Table 2. Comparison with representative OOD detection methods in hard OOD detection on the CIFAR-10 dataset.

CIFAR-10	ImageNet-Fix		ImageNet-Resize		Oxford-Pets		Average	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
MSP	32.58	89.09	22.72	92.02	100.00	63.56	51.77	81.56
MaxLogit	33.47	91.40	13.87	96.07	90.49	60.11	61.98	82.53
ODIN	33.37	91.36	11.55	96.99	91.43	59.38	45.45	82.58
Mahalanobis	86.37	61.02	65.81	82.20	89.35	57.38	80.51	66.87
Energy	33.49	91.41	13.60	96.14	90.49	60.06	45.86	82.54
GradNorm	33.33	91.40	13.86	96.07	91.67	58.47	46.29	81.98
ReAct	63.51	76.95	25.83	91.96	94.80	40.35	61.38	69.75
DICE	42.62	88.88	12.61	97.22	89.63	59.56	48.29	81.89
ASH	32.69	91.49	13.69	96.14	90.64	59.61	45.67	82.41
InsRect	**30.70**	**91.83**	**10.90**	**96.88**	**86.52**	**63.51**	**42.71**	**84.07**

Table 3. Comparison with representative OOD detection methods in hard OOD detection on the CIFAR-100 dataset.

CIFAR-100	ImageNet-Fix		ImageNet-Resize		Oxford-Pets		Average	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
MSP	70.25	73.35	77.40	65.36	59.17	76.98	68.94	71.90
MaxLogit	70.25	77.44	75.19	72.75	60.58	77.55	68.67	75.91
ODIN	70.29	77.70	70.53	76.16	62.31	76.83	67.71	76.90
Mahalanobis	87.62	60.67	37.52	92.74	86.71	54.68	70.62	69.36
Energy	70.10	77.42	75.24	73.01	60.87	77.21	68.74	75.88
GradNorm	70.21	77.46	74.87	73.17	62.70	75.99	69.26	75.54
ReAct	74.73	70.71	35.20	88.61	60.17	76.76	56.70	78.69
DICE	73.14	75.10	46.12	87.51	83.04	59.83	67.43	74.15
ASH	69.31	78.08	58.65	79.52	64.63	75.48	64.20	77.69
InsRect	**66.67**	**78.59**	**17.80**	**95.52**	**51.90**	**81.23**	**45.56**	**85.11**

Table 4. Comparison with representative OOD detection methods on CIFAR-100 with backbones of ResNet-50 and WRN-40-2.

CIFAR-100	ResNet-50		WRN-40-2	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
MSP	58.66	78.03	67.34	74.90
MaxLogit	54.99	81.26	64.92	78.08
ODIN	54.32	81.75	65.74	77.82
Mahalanobis	50.43	81.87	61.39	75.41
Energy	54.78	81.66	64.85	78.09
GradNorm	54.32	81.86	63.93	78.38
ReAct	47.18	84.70	79.86	63.36
DICE	54.71	82.47	66.36	75.89
ASH	61.09	81.06	69.16	75.56
InsRect	**42.98**	**85.54**	**45.67**	**85.51**

Table 5. Comparison with representative OOD detection methods on ImageNet-1K with backbones of WRN-50-2 and DenseNet-161.

ImageNet-1K	WRN-50-2		DenseNet-161	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
MSP	54.37	84.19	61.39	80.67
MaxLogit	50.34	86.59	58.17	84.88
ODIN	48.68	87.30	59.21	84.75
Mahalanobis	55.55	78.66	75.29	63.87
Energy	50.45	86.21	58.31	84.77
GradNorm	63.97	77.06	68.17	77.74
ReAct	40.20	88.11	43.70	87.88
DICE	87.44	53.00	74.00	80.12
ASH	38.94	91.12	46.20	89.73
InsRect	**31.21**	**92.63**	**37.79**	**92.03**

search costs for thresholding, so we prefer relatively small J, especially for sample tasks such as CIFAR-10.

Number of Searching Iterations. We fix the value of T to be 500 for BO iterations, while other values of T may also be of our interests. Table 8 summarizes the results on the CIFAR benchmark with different values of T. As we can see, choosing $T = 100 \sim 200$ has achieved attractive performance across different setups, and the performance remains stable with much more searching steps.

Methods for Common Components. Besides NMF, both PCA and ICA can also be used to define common components. Thus, we conduct the experiments in comparing between PCA, NMF and ICA, summarizing the results in

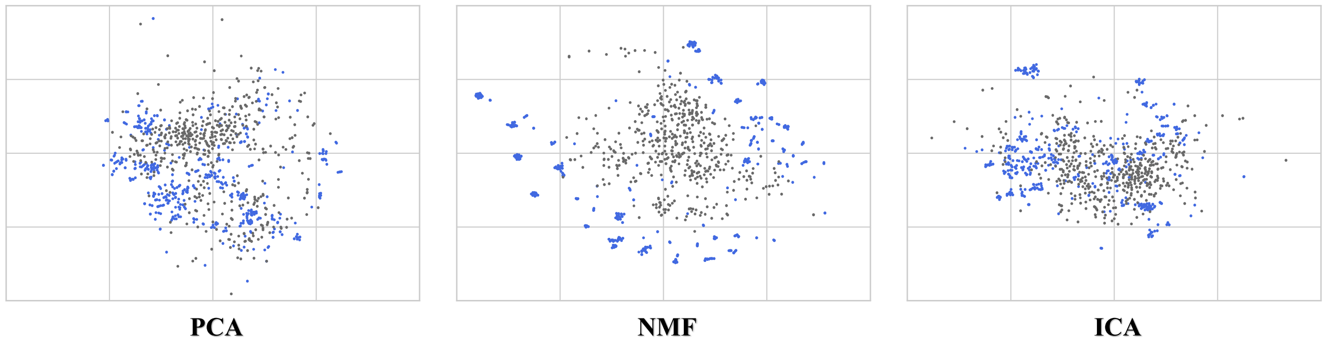


Figure 3. Visualization of embedding features after InsRect, across varying ways in defining common components. The blue points refer to ID data (CIFAR-100) and grey points refer to OOD data (Places365). As we can see, InsRect with NMF can better separate ID and OOD data in the embedding space, thus leading to improved OOD performance.

Table 6. Comparison between different methods in generating common components.

Algorithms	CIFAR-10		CIFAR-100	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
PCA	48.00	79.67	41.54	87.51
NMF	**11.98**	**97.36**	**27.52**	**92.88**
ICA	48.33	84.76	47.19	86.68

Table 7. The performance of OOD detection with different number of the common components varying from 5 to 100.

Components	CIFAR-10		CIFAR-100	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
5	12.50	97.30	46.68	83.57
10	**11.98**	**97.36**	38.09	87.44
20	12.60	97.30	32.09	90.87
50	12.13	97.32	**27.52**	**92.88**
80	12.19	97.31	27.65	92.83
100	12.24	97.31	28.38	92.78

Table 8. The performance of OOD detection with different number of the iterations varying from 10 to 500.

Iterations	CIFAR-10		CIFAR-100	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
10	12.94	97.21	39.68	88.68
20	12.12	97.32	39.68	88.68
50	12.14	97.27	36.76	89.52
100	**11.98**	**97.36**	32.40	90.91
200	**11.98**	**97.36**	27.81	92.69
500	**11.98**	**97.36**	**27.52**	**92.88**

Table 6. Moreover, we depict the associated t-SNE visualization in Figure 3 for the resultant embedding features.

Table 9. The effects of adopting bottom thresholds and upper thresholds, comparing with the base form of score (Energy) adopted by InsRect.

Thresholds	CIFAR-10		CIFAR-100	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
w/o r, a	20.98	94.52	67.88	75.22
w/o r	12.35	97.28	41.35	87.43
w/o a	12.26	97.34	27.94	92.73
InsRect	**11.98**	**97.36**	**27.52**	**92.88**

From both evaluation and visualization, NMF is superior to other methods in finding common components, validating our design choice in using NMF.

Bottom and Upper Thresholds. By default, we use both bottom and upper thresholds for common components. To further demonstrate their respective power, we conduct experiments in Table 9, comparing the results without thresholds (w/o r, a), without upper thresholds (w/o r), without bottom thresholds (w/o a), and with both bottom and upper thresholds (InsRect). As we can see, both bottom and upper thresholds contribute, while upper thresholds are more important, especially for the CIFAR-100 setup.

5. Conclusion

In the paper, we propose a novel framework of features rectification named InsRect, clamping common components instead of original embedding features as in previous works. Our method provides a more flexible and effective manner over existing techniques, further facilitating instance-customized rectification that are overlooked in previous works. We suggest BO for automatic hyper-parameter search, which is superior to manual tuning especially for large J. In the future, we will explore more parameter-effective rules of features rectification and propose more general and flexible methods to boost OOD detection. The discussion of our limitations is available in Appendix F.

References

[1] Abhijit Bendale and Terrance E. Boult. Towards open set deep networks. In *CVPR*, 2016. 1

[2] Saikiran Bulusu, Bhavya Kailkhura, Bo Li, P Varshney, and Dawn Song. Anomalous instance detection in deep learning: A survey. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2020. 1

[3] Jiefeng Chen, Yixuan Li, Xi Wu, Yingyu Liang, and Somesh Jha. ATOM: robustifying out-of-distribution detection using outlier mining. In *ECML*, 2021. 1, 2

[4] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 5

[5] Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–240. Springer, 2013. 4

[6] Jesse Davis and Mark Goadrich. The relationship between precision-recall and ROC curves. In *ICML*, 2006. 5

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5, 6

[8] Andrija Djurisic, Nebojsa Bozanic, Arjun Ashok, and Rosanne Liu. Extremely simple activation shaping for out-of-distribution detection. *arXiv preprint arXiv:2209.09858*, 2022. 1, 3, 4, 5, 2

[9] Xuefeng Du, Gabriel Gozum, Yifei Ming, and Yixuan Li. Siren: Shaping representations for detecting out-of-distribution objects. In *NeurIPS*, 2022. 2

[10] Xuefeng Du, Zhaoning Wang, Mu Cai, and Yixuan Li. VOS: learning what you don't know by virtual outlier synthesis. In *ICLR*, 2022. 1, 2

[11] Zhen Fang, Yixuan Li, Jie Lu, Jiahua Dong, Bo Han, and Feng Liu. Is out-of-distribution detection learnable? In *NeurIPS*, 2022. 1

[12] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *ICML*, 2017. 1, 3

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 6

[14] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017. 1, 5, 2

[15] Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. Deep anomaly detection with outlier exposure. In *ICLR*, 2019. 1, 5, 2

[16] 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 *ICML*, 2022. 1, 5

[17] Jens Henriksson, Christian Berger, Markus Borg, Lars Tornberg, Sankar Raman Sathyamoorthy, and Cristofer Englund. Performance analysis of out-of-distribution detection on trained neural networks. *Information and Software Technology*, 130:106409, 2021. 6

[18] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alexander Shepard, Hartwig Adam, Pietro Perona, and Serge J. Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018. 5

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 5, 6

[20] Haiwen Huang, Zhihan Li, Lulu Wang, Sishuo Chen, Bin Dong, and Xinyu Zhou. Feature space singularity for out-of-distribution detection. *arXiv preprint arXiv:2011.14654*, 2020. 2

[21] Rui Huang, Andrew Geng, and Yixuan Li. On the importance of gradients for detecting distributional shifts in the wild. In *NeurIPS*, 2021. 5, 2

[22] Conor Igoe, Youngseog Chung, Ian Char, and Jeff Schneider. How useful are gradients for ood detection really? In *arXiv preprint arXiv:2205.10439*, 2022. 2

[23] Julian Katz-Samuels, Julia B Nakhleh, Robert Nowak, and Yixuan Li. Training ood detectors in their natural habitats. In *ICML*, 2022. 2, 5

[24] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical Report TR-2009, University of Toronto*, 2009. 5

[25] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 5

[26] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *ICLR*, 2018. 1, 2

[27] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*, 2018. 1, 5, 2

[28] Seung Lee. Learning the parts of objects by non-negative matrix factorization. *Nature*, pages 788–791, 1999. 2

[29] Yi Li and Nuno Vasconcelos. Background data resampling for outlier-aware classification. In *CVPR*, 2020. 2

[30] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017. 1, 5, 2

[31] Weitang Liu, Xiaoyun Wang, John D Owens, and Yixuan Li. Energy-based out-of-distribution detection. In *NeurIPS*, 2020. 1, 3, 5, 2

[32] Yifei Ming, Ying Fan, and Yixuan Li. POEM: out-of-distribution detection with posterior sampling. In *ICML*, 2022. 1, 2

[33] Yifei Ming, Yiyou Sun, Ousmane Dia, and Yixuan Li. How to exploit hyperspherical embeddings for out-of-distribution detection? In *ICLR*, 2023. 2

[34] Peyman Morteza and Yixuan Li. Provable guarantees for understanding out-of-distribution detection. In *AAAI*, 2022. 2

[35] Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Görür, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *ICLR*, 2019. 1

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

[37] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, pages 427–436, 2015. 1

[38] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar. Cats and dogs. In *CVPR*, 2012. 6

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 5

[40] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021. 5

[41] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 5

[42] Eric Schulz, Maarten Speekenbrink, and Andreas Krause. A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, 2018. 4

[43] Vikash Sehwag, Mung Chiang, and Prateek Mittal. SSD: A unified framework for self-supervised outlier detection. In *ICLR*, 2021. 1, 2

[44] Jonathon Shlens. A tutorial on independent component analysis. *arXiv preprint arXiv:1404.2986*, 2014. 2

[45] Deva Ramanan Shu Kong. Opengan: Open-set recognition via open data generation. In *ICCV*, 2021. 5

[46] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *NIPS*, 25, 2012. 2

[47] Yue Song, Nicu Sebe, and Wei Wang. Rankfeat: Rank-1 feature removal for out-of-distribution detection. *NeurIPS*, 35:17885–17898, 2022. 1, 2

[48] Yiyou Sun and Yixuan Li. Dice: Leveraging sparsification for out-of-distribution detection. In *ECCV*, pages 691–708. Springer, 2022. 5

[49] Yiyou Sun, Chuan Guo, and Yixuan Li. React: Out-of-distribution detection with rectified activations. *NeurIPS*, 34: 144–157, 2021. 1, 4, 5, 2

[50] Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. In *ICML*, 2022. 1

[51] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 3

[52] Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. CSI: novelty detection via contrastive learning on distributionally shifted instances. In *NeurIPS*, 2020. 1, 6, 2

[53] Leitian Tao, Xuefeng Du, Xiaojin Zhu, and Yixuan Li. Non-parametric outlier synthesis. *arXiv preprint arXiv:2303.02966*, 2023. 2

[54] Haoran Wang, Weitang Liu, Alex Bocchieri, and Yixuan Li. Can multi-label classification networks know what they don't know? In *NeurIPS*, 2021. 2

[55] Qizhou Wang, Feng Liu, Yonggang Zhang, Jing Zhang, Chen Gong, Tongliang Liu, and Bo Han. Watermarking for out-of-distribution detection. In *NeurIPS*, 2022. 1, 2

[56] Qizhou Wang, Zhen Fang, Yonggang Zhang, Feng Liu, Yixuan Li, and Bo Han. Learning to augment distributions for out-of-distribution detection. *arXiv preprint arXiv:2311.01796*, 2023. 2

[57] Qizhou Wang, Junjie Ye, Feng Liu, Quanyu Dai, Marcus Kalander, Tongliang Liu, Jianye Hao, and Bo Han. Out-of-distribution detection with implicit outlier transformation. In *ICLR*, 2023. 1, 5, 2

[58] Hongxin Wei, Renchunzi Xie, Hao Cheng, Lei Feng, Bo An, and Yixuan Li. Mitigating neural network overconfidence with logit normalization. In *ICML*, 2022. 2

[59] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987. 2

[60] Xiaobo Xia, Tongliang Liu, Bo Han, Nannan Wang, Mingming Gong, Haifeng Liu, Gang Niu, Dacheng Tao, and Masashi Sugiyama. Part-dependent label noise: Towards instance-dependent label noise. *NeurIPS*, 33:7597–7610, 2020. 3

[61] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010. 5

[62] Pingmei Xu, Krista A Ehinger, Yinda Zhang, Adam Finkelstein, Sanjeev R Kulkarni, and Jianxiong Xiao. Turkergaze: crowdsourcing saliency with webcam based eye tracking. *arXiv preprint arXiv:1504.06755*, 2015. 5

[63] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021. 1

[64] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 5

[65] Alireza Zaeemzadeh, Niccolò Bisagno, Zeno Sambugaro, Nicola Conci, Nazanin Rahnavard, and Mubarak Shah. Out-of-distribution detection using union of 1-dimensional subspaces. In *CVPR*, 2021. 2

[66] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 6

[67] Bolei Zhou, Àgata 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, 2018. 5

Data Specified Feature Rectification for Out-of-distribution Detection

Supplementary Material

Contents

A. Related works

Recent advances in OOD detection can be broadly attributed into two categories: fine-tuning approaches and post-hoc approaches. We briefly review these two lines of works as follows.

Fine-tuning approaches. Fine-tuning approaches adjust model parameters to improve OOD performance, where existing works are attributed into two categories, i.e., representation-based methods [52] and outlier exposure [15]. Specifically, the representation-based approaches aim at devising advanced learning strategies to pursue improved representation that can facilitate OOD detection. Existing works use off-the-shelf contrastive learning methods [43, 52], constrain on manifold structures for embedding features [9, 33, 65] and employ various regularization strategies [58]. On the other side, for outlier exposure, we leverage a set of auxiliary OOD datasets during training, making models learn to discern ID and OOD patterns directly. Advanced works focus on mining informative OOD examples [3, 29, 32] and enlarging the coverage of auxiliary OOD data [10, 26, 53, 56, 57]. Other works also study the cases where some OOD data contain ID semantics, focusing on robust outlier exposure [23]. However, fine-tuning approaches require model re-training, of which the costs can be prohibitively high for many real-world applications.

Post-hoc approaches. By contrast, other researchers believe that conventionally trained models, leveraging cross entropy loss on ID data, can already excel at OOD detection. Therefore, post-hoc approaches typically assume model parameters cannot be further tuned, leveraging model responses to design various scoring functions that can discern ID/OOD. Overall, a proper scoring function should discern features between ID and OOD cases, and existing works exploit such features discrepancy in the logit space [14, 30, 31, 34, 54, 55], the embedding space [8, 20, 27, 47, 49, 65] and the gradient space [21, 22]. Recent works also find that there exist some features that are misleading for OOD detection, thus suggesting features rectifications [8, 47, 49] to mitigate their negative impacts. They typically assume extremely small (or large) features elements are misleading, suggesting to prune (or truncate) such features [8, 49]. However, their forms of rectification are very limited, largely hindering their potential power. It motivates us to study more flexible forms of features rectification that can generalize and improve previous works.

B. Hardware Configurations

All experiments are conducted in Pytorch 1.11.0 with CUDA 11.3, where the machine is equipped with a series of GeForce RTX 3090 GPUs and AMD EPYC 7642 48-Core Processors.

C. Pre-training Setups

For the CIFAR benchmark, the models have been trained for 200 epochs via cross-entropy loss minimization, with the batch size 16, momentum 0.9, and initial learning rate 0.1. The learning rate is divided by 5 after 60, 120 and 160 epochs. For the ImageNet-1K benchmark, we adopt the pretrained parameters provided by PyTorch following [8].

D. Full Evaluation Results

The average performance on the CIFAR and ImageNet-1K benchmarks are reported in the main text, and we further provide a more detailed results across individual test OOD datasets, summarizing the results in Tables 10-12. As we can see, not only the average performance of InsRect are higher than previous works in most cases, that of individual test OOD datasets are also superior or comparable to the baselines in most cases. It demonstrates the effectiveness of our InsRect across different OOD cases.

E. Bayesian Optimization

The primary idea of BO is to estimate the optimization objective $\mathcal{M}(\{a_j\}_j, \{r_j\}_j)$ with a probabilistic model of $G\big(\mathcal{M}(\{a_j\}_j, \{r_j\}_j)|g(\{a_j\}_j, \{r_j\}_j)\big)$, which plays the role of guiding our sampling toward optimal adaptation. BO consists of two main parts, namely, the surrogate function and the acquisition protocol. The surrogate function is a probabilistic model for modeling the unknown function, which can estimate the mean and variance of unknown function. The acquisition protocol is a scalar function based on the surrogate function, with the goal of finding a balance between exploration and exploitation. Thus, acquisition protocol needs to consider both the optimal result in explored region and the uncertainty in unexplored region. Overall, the surrogate function serves a dual purpose in BO: the probabilistic model guides the sampling procedure (with the acquisition protocol) to find better parameters, and the sampled results in turn, improve its surrogate estimation. Specifically, BO consists of five steps in each iteration t:

Table 10. The detailed results (include six common OOD datasets on the CIFAR benchmark) of comparison with representative OOD detection methods on the CIFAR-10. $\downarrow$ (or $\uparrow$) indicates smaller (or larger) values are preferred. The best result in each column is indicated in bold.

Model	Methods	Textures		Places365		LSUN-Crop		LSUN-Resize		iSUN		SVHN		Average	
		FPR95 $\downarrow$	AUROC $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$
ResNet-50	MSP	45.40	87.78	32.93	90.43	16.09	95.05	28.21	91.65	35.63	89.95	60.56	80.64	36.47	89.25
	MaxLogit	60.19	86.84	31.66	92.36	8.41	98.04	23.83	94.29	30.74	92.95	59.15	81.87	35.66	91.06
	ODIN	62.97	86.21	34.36	91.94	7.68	98.19	22.27	94.96	27.74	93.76	61.32	80.58	36.06	90.94
	Mahalanobis	**38.12**	**91.23**	60.43	73.74	33.35	86.14	48.17	83.25	51.22	81.48	27.65	90.17	43.16	84.34
	Energy	60.20	86.84	31.66	92.45	8.07	98.21	23.79	94.43	30.72	93.09	59.15	81.90	35.60	91.15
	GradNorm	72.68	83.40	32.32	92.31	8.30	98.07	23.24	94.47	31.83	92.73	41.61	88.08	35.00	91.51
	ReAct	58.94	86.99	31.58	92.45	8.12	98.21	23.15	94.53	29.62	93.22	59.51	81.84	35.15	91.21
	DICE	57.48	87.39	32.32	**92.66**	**4.11**	**99.18**	37.83	91.13	41.12	90.51	**27.07**	**91.92**	33.32	92.13
	ASH	64.36	87.88	39.71	91.63	5.57	98.76	20.29	95.00	**22.31**	**94.40**	53.93	84.16	34.36	91.97
	InsRect	58.18	88.20	**31.08**	92.65	6.51	98.53	**19.58**	**95.17**	24.07	94.23	53.31	84.25	**32.12**	**92.17**
WRN-40-2	MSP	56.98	85.50	**44.84**	87.83	20.19	93.65	33.96	90.25	43.97	88.14	45.69	85.99	**40.94**	**88.56**
	MaxLogit	83.27	81.18	56.89	88.69	13.39	96.94	33.42	92.85	44.64	90.82	78.16	75.95	51.63	87.74
	ODIN	86.37	79.51	61.38	87.58	13.88	96.91	34.32	93.03	44.16	91.04	82.29	73.21	53.73	86.88
	Mahalanobis	**37.20**	**91.29**	66.25	72.58	49.83	81.34	64.35	74.78	64.47	74.16	**23.90**	**94.92**	51.00	81.51
	Energy	83.27	81.01	56.88	**88.75**	13.03	97.10	33.30	92.98	44.59	90.95	78.16	75.45	51.54	87.71
	GradNorm	88.93	77.20	57.33	88.60	13.29	96.99	32.60	93.08	45.52	90.58	61.26	83.66	49.82	88.35
	ReAct	60.27	80.91	69.25	81.75	59.88	82.56	23.91	**94.88**	33.54	**92.95**	72.94	63.11	53.30	82.69
	DICE	89.94	74.19	67.55	85.25	**8.40**	**98.37**	43.55	91.16	51.43	89.36	87.93	66.50	58.13	84.14
	ASH	89.04	80.02	72.31	84.52	12.99	97.19	39.51	92.42	47.53	90.39	90.63	72.49	58.67	86.17
	InsRect	54.27	81.46	62.04	79.79	17.69	89.39	**23.19**	88.46	**26.25**	88.03	63.85	74.59	41.22	83.62
DenseNet-161	MSP	39.23	87.25	31.17	89.75	16.81	94.13	18.92	93.38	19.29	93.35	17.61	93.94	23.84	91.96
	MaxLogit	57.97	85.51	26.43	93.38	7.00	98.34	9.46	97.41	9.55	97.34	16.14	95.10	21.09	94.51
	ODIN	57.80	85.81	27.55	93.29	7.09	98.42	**7.18**	**98.13**	**7.41**	98.08	17.85	94.70	20.81	94.74
	Mahalanobis	51.08	90.66	89.45	53.40	64.56	77.05	61.76	80.56	63.12	82.36	18.42	96.73	58.07	80.13
	Energy	58.05	85.41	26.52	93.43	6.67	98.43	9.22	97.48	9.33	97.42	16.08	94.98	20.98	94.52
	GradNorm	62.18	83.57	26.47	93.39	6.96	98.36	9.43	97.43	9.95	97.21	11.81	96.70	21.13	94.44
	ReAct	37.73	89.63	35.51	91.04	15.44	96.22	11.66	97.10	13.03	96.74	15.90	94.24	21.55	94.16
	DICE	65.78	86.53	25.95	94.01	1.73	99.61	8.47	98.10	8.47	98.16	16.73	96.53	21.19	95.49
	ASH	45.22	89.23	**22.74**	**94.35**	5.20	98.85	8.56	97.67	8.16	97.74	9.58	97.25	16.58	95.85
	InsRect	**20.68**	**95.63**	26.27	93.80	**3.53**	**99.31**	9.19	98.02	7.93	**98.28**	**4.27**	**99.10**	**11.98**	**97.36**

Table 11. The detailed results (include six common OOD datasets on the CIFAR benchmark) of comparison with representative OOD detection methods on the CIFAR-100. $\downarrow$ (or $\uparrow$) indicates smaller (or larger) values are preferred. The best result in each column is indicated in bold.

Model	Methods	Textures		Places365		LSUN-Crop		LSUN-Resize		iSUN		SVHN		Average	
		FPR95 $\downarrow$	AUROC $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$
ResNet-50	MSP	63.63	76.54	77.53	72.00	43.52	84.18	56.23	78.98	57.69	78.09	53.38	78.38	58.66	78.03
	MaxLogit	62.92	78.69	80.95	71.48	34.86	88.72	51.31	83.28	52.59	82.75	47.30	82.64	54.99	81.26
	ODIN	62.94	79.23	81.48	70.95	34.10	89.26	49.15	84.63	50.50	84.24	47.75	82.22	54.32	81.75
	Mahalanobis	**28.21**	**94.86**	81.44	71.86	80.43	57.14	39.93	87.47	40.39	88.00	32.19	**91.92**	50.43	81.87
	Energy	63.05	78.85	80.95	71.42	34.15	89.35	51.14	83.83	52.36	83.35	47.04	83.13	54.78	81.66
	GradNorm	69.58	74.69	80.78	71.58	34.01	89.42	51.10	83.77	53.41	82.56	37.04	89.15	54.32	81.86
	ReAct	53.31	82.86	83.71	70.61	32.35	90.21	37.34	89.04	38.07	88.79	38.30	86.71	47.18	84.70
	DICE	65.21	79.29	82.63	69.65	28.77	**93.32**	56.21	82.41	56.74	82.20	38.70	87.93	54.71	82.47
	ASH	51.80	87.75	81.48	71.00	**28.75**	93.24	89.48	69.93	85.22	72.78	**29.83**	91.66	61.09	81.06
	InsRect	51.19	83.64	**70.41**	**73.68**	29.05	90.78	**33.51**	**89.59**	**35.83**	**88.84**	37.88	86.73	**42.98**	**85.54**
WRN-40-2	MSP	73.67	71.52	76.46	70.10	62.15	80.69	65.54	75.98	69.17	74.01	57.06	77.11	67.34	74.90
	MaxLogit	75.84	73.26	76.93	72.04	42.49	89.96	67.10	78.05	70.48	75.60	56.67	79.57	64.92	78.08
	ODIN	77.61	72.47	77.96	71.34	42.60	89.96	67.19	78.92	69.56	76.57	59.49	77.62	65.74	77.82
	Mahalanobis	**41.91**	**90.51**	84.88	63.97	89.21	43.45	48.61	85.62	51.12	85.74	52.61	83.18	61.39	75.41
	Energy	75.96	73.12	77.11	71.99	41.36	90.70	67.16	77.89	70.46	75.41	57.05	79.45	64.85	78.09
	GradNorm	79.59	69.25	76.80	72.16	41.68	90.51	67.46	77.99	71.04	74.77	47.00	85.58	63.93	78.38
	ReAct	85.93	59.50	83.67	65.90	72.70	72.12	76.76	66.94	80.08	63.93	80.01	51.76	79.86	63.36
	DICE	80.27	70.43	77.02	70.35	33.38	93.81	73.60	71.40	75.01	69.82	58.87	79.50	66.36	75.89
	ASH	76.50	78.20	84.75	65.07	30.50	**93.66**	85.72	64.48	84.05	66.63	53.46	85.34	69.16	75.56
	InsRect	57.45	82.78	**65.73**	**77.67**	**29.95**	91.80	**41.84**	**86.63**	**39.98**	**87.20**	**39.06**	**87.01**	**45.67**	**85.51**
DenseNet-161	MSP	78.16	70.55	76.12	69.58	64.36	76.36	85.59	56.93	82.53	60.45	66.88	71.89	75.61	67.63
	MaxLogit	93.36	70.37	69.59	74.65	45.04	86.94	77.51	68.76	77.71	69.39	45.00	80.28	68.04	75.06
	ODIN	91.61	71.37	69.23	75.09	44.39	87.35	73.56	72.29	73.90	72.97	46.80	78.74	66.58	76.30
	Mahalanobis	**48.03**	**91.42**	93.81	53.01	88.05	54.56	30.10	93.46	35.34	92.80	55.32	84.88	58.44	78.35
	Energy	93.34	70.22	69.58	74.69	44.70	87.27	77.40	69.16	77.60	69.68	44.66	80.33	67.88	75.22
	GradNorm	94.46	67.89	69.25	74.84	44.68	87.24	77.08	69.29	77.70	69.36	38.97	84.79	67.02	75.57
	ReAct	80.42	64.33	91.75	56.81	75.15	60.95	47.37	82.86	56.98	78.84	65.74	64.92	69.57	68.12
	DICE	92.01	77.29	58.74	81.51	24.64	95.34	55.60	83.50	54.68	85.02	35.68	90.16	53.56	85.47
	ASH	48.49	89.42	57.75	80.41	23.56	94.71	50.22	81.55	45.50	84.57	**20.36**	**94.91**	40.98	87.59
	InsRect	55.32	87.48	**54.51**	**84.19**	**14.93**	**96.58**	**9.82**	**97.55**	**9.24**	**97.82**	21.28	93.64	**27.52**	**92.88**

- Rectify common components with $\{a_j^{(t)}\}_j, \{r_j^{(t)}\}_j$ in Eq. (7);
- Reconstruct features in Eq. (8);
- Compute scores of ID (training) and OOD (auxiliary) data in Eq. (9);

Table 12. The detailed results (include four common OOD datasets on the ImageNet-1K benchmark) of comparison with representative OOD detection methods on the ImageNet-1K. ↓ (or ↑) indicates smaller (or larger) values are preferred. The best result in each column is indicated in bold.

Model	Methods	Textures		Places365		iNaturalist		SUN		Average	
		FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
ResNet-50	MSP	67.76	80.43	58.23	80.54	43.38	88.39	58.59	81.64	56.99	82.75
	MaxLogit	56.22	86.39	55.64	84.03	30.60	91.15	49.78	86.44	48.06	87.00
	ODIN	53.98	87.80	55.92	84.11	29.92	92.24	50.41	86.76	47.56	87.73
	Mahalanobis	31.52	93.10	81.47	59.80	67.47	69.71	80.30	60.48	65.19	70.77
	Energy	56.40	86.72	55.81	83.96	31.31	90.62	49.70	86.57	48.31	86.97
	GradNorm	73.17	75.43	68.42	75.17	44.48	83.21	62.97	78.30	62.26	78.03
	ReAct	37.36	90.31	40.14	91.97	16.93	96.40	28.08	94.46	30.63	93.29
	DICE	51.95	90.24	76.61	78.20	54.08	87.58	68.01	83.85	62.66	84.97
	ASH	15.61	97.13	39.72	90.86	10.97	97.73	29.34	93.73	23.91	94.86
	InsRect	**14.92**	**97.17**	**35.50**	**92.89**	**5.51**	**98.82**	**24.31**	**95.27**	**20.06**	**96.04**
MobileNetV2	MSP	68.61	78.93	63.56	78.10	46.33	86.72	64.05	78.81	60.64	80.64
	MaxLogit	55.37	86.07	56.42	83.01	30.57	90.68	49.54	85.71	47.98	86.37
	ODIN	53.23	87.35	56.69	83.10	30.10	91.64	50.15	85.95	47.54	87.01
	Mahalanobis	80.75	77.80	99.42	31.57	98.47	33.17	99.50	28.23	94.54	42.69
	Energy	54.39	86.57	56.61	83.15	31.26	90.37	49.36	86.17	47.91	86.56
	GradNorm	70.34	75.34	68.18	74.18	43.61	82.42	61.54	77.54	60.91	77.37
	ReAct	41.32	90.96	56.99	84.04	**26.89**	92.76	48.96	87.21	43.54	88.74
	DICE	42.55	91.75	66.74	82.49	50.66	88.48	51.49	88.78	52.86	87.87
	ASH	30.86	93.69	50.49	85.32	26.97	92.84	42.17	88.95	37.62	90.20
	InsRect	**23.61**	**95.49**	**47.93**	**86.86**	28.13	**93.23**	**38.61**	**90.64**	**34.57**	**91.55**
WRN-50-2	MSP	66.86	81.08	55.88	82.29	39.00	89.99	55.74	83.38	54.37	84.19
	MaxLogit	63.99	84.22	55.24	84.50	31.06	91.31	51.07	86.33	50.34	86.59
	ODIN	59.81	85.52	55.25	84.50	28.35	92.57	51.32	86.60	48.68	87.30
	Mahalanobis	**38.25**	91.34	68.35	70.88	53.12	79.98	62.48	72.43	55.55	78.66
	Energy	64.27	84.09	55.22	84.20	31.38	90.43	50.93	86.11	50.45	86.21
	GradNorm	78.46	72.10	68.02	75.28	45.03	83.24	64.37	77.62	63.97	77.06
	ReAct	69.40	77.62	37.55	89.76	26.91	92.60	26.94	92.48	40.20	88.11
	DICE	82.42	63.31	92.26	45.89	85.07	53.28	90.02	49.54	87.44	53.00
	ASH	41.87	**91.58**	49.71	87.47	18.55	95.84	45.61	89.59	38.94	91.12
	InsRect	44.35	88.79	**36.84**	**91.60**	**12.49**	**97.12**	**31.15**	**93.01**	**31.21**	**92.63**
DenseNet-161	MSP	77.10	76.25	60.60	79.51	44.66	87.20	63.20	79.71	61.39	80.67
	MaxLogit	78.06	81.24	63.61	82.00	29.68	92.14	61.35	84.15	58.17	84.88
	ODIN	76.53	81.76	68.01	80.61	27.77	93.35	64.51	83.28	59.21	84.75
	Mahalanobis	45.12	90.09	89.55	53.33	77.40	60.30	89.08	51.75	75.29	63.87
	Energy	78.04	81.26	63.88	81.79	29.76	91.94	61.58	84.10	58.31	84.77
	GradNorm	86.98	72.00	73.39	74.77	40.86	86.67	71.47	77.53	68.17	77.74
	ReAct	**35.62**	**90.96**	60.77	82.24	30.69	91.83	47.72	86.48	43.70	87.88
	DICE	88.80	72.86	78.84	77.43	62.61	85.71	65.75	84.47	74.00	80.12
	ASH	66.37	85.78	50.94	87.45	20.95	95.51	46.52	90.18	46.20	89.73
	InsRect	45.06	90.49	**47.46**	**89.23**	**19.92**	**96.20**	**38.72**	**92.21**	**37.79**	**92.03**

- Compute optimization objective (FPR95) with the scores;
- Update the optimal set of clamping thresholds and the optimal value of optimization objective.

We repeat steps 1-5 for T iterations to meet the stopping criterion, and the overall framework in BO optimization is summarized in Algorithm 2. Moreover, in our realization, we adopt the *Gaussian Process* (GP) [42] as the surrogate function and the *Upper Confidence Bound* (UCB) [5] as the acquisition protocol, which are commonly used in the community.

Surrogate Function. For convenience of expression, here we denote the set of clamping thresholds $\{a_j\}_j, \{r_j\}_j$ by the variable $\mathbf{v}$, referring to a data point in the optimization space. In the initialization of BO, we can obtain the values of optimization objective $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$ for some data points $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$, where $y_i = g(\mathbf{v}_i) + \eta_i$, $\eta_i \sim \mathbf{N}(0, \sigma_n^2)$, $\sigma_n^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - g(\mathbf{v}_i))^2$. GP assumes that the unknown function to estimate has the form as follows.

$$\begin{pmatrix} g(\mathbf{v}_1) \\ g(\mathbf{v}_2) \\ \vdots \\ g(\mathbf{v}_n) \end{pmatrix} \sim \mathbf{N}\left(\begin{pmatrix} m(\mathbf{v}_1) \\ m(\mathbf{v}_2) \\ \vdots \\ m(\mathbf{v}_n) \end{pmatrix}, \begin{pmatrix} k(\mathbf{v}_1, \mathbf{v}_1) & k(\mathbf{v}_1, \mathbf{v}_2) & \cdots & k(\mathbf{v}_1, \mathbf{v}_n) \\ k(\mathbf{v}_2, \mathbf{v}_1) & k(\mathbf{v}_2, \mathbf{v}_2) & \cdots & k(\mathbf{v}_2, \mathbf{v}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{v}_n, \mathbf{v}_1) & k(\mathbf{v}_n, \mathbf{v}_2) & \cdots & k(\mathbf{v}_n, \mathbf{v}_n) \end{pmatrix}\right) \tag{17}$$

Algorithm 2 Search for clamping thresholds with BO

```
Require: ID (training) and OOD (auxiliary) data $\{\mathbf{x}_k^{\mathrm{I}}\}_k, \{\mathbf{x}_k^{\mathrm{A}}\}_k$, and common components $\{\mathbf{c}_j\}_j$.
1: for $t = 1 \dots T$ do
2:     Sample clamping thresholds $\{a_j^{(t)}\}_j, \{r_j^{(t)}\}_j$ by acquisition protocol;
3:     Compute $\mathcal{M}(\{a_j^{(t)}\}_j, \{r_j^{(t)}\}_j)$ with ID and OOD data;
4:     if $\mathcal{M}(\{a_j^{(t)}\}_j, \{r_j^{(t)}\}_j) < \mathcal{M}(\{a_j^{(*)}\}_j, \{r_j^{(*)}\}_j)$ then
5:         Update $\mathcal{M}(\{a_j^{(*)}\}_j, \{r_j^{(*)}\}_j) \leftarrow \mathcal{M}(\{a_j^{(t)}\}_j, \{r_j^{(t)}\}_j)$
6:         Update $\{a_j^{(*)}\}_j, \{r_j^{(*)}\}_j \leftarrow \{a_j^{(t)}\}_j, \{r_j^{(t)}\}_j$
7:     end if
8:     Update surrogate function;
9: end for
10: return $\{a_j^{(*)}\}_j, \{r_j^{(*)}\}_j$.
```

where $m(\cdot)$ is the function of mean, $k(\cdot,\cdot)$ is the function of covariance, and $\mathbf{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.
Acquisition Protocol. The acquisition protocol adopting UCB can be define as follows

$$A(\{a_j\}_j, \{r_j\}_j) = \mu(\{a_j\}_j, \{r_j\}_j) + \kappa\sigma(\{a_j\}_j, \{r_j\}_j) \tag{18}$$

where $\{a_j\}_j, \{r_j\}_j$ are parameters (clamping thresholds), $\mu(\{a_j\}_j, \{r_j\}_j)$ and $\sigma(\{a_j\}_j, \{r_j\}_j)$ are the mean and standard deviation of the surrogate function, respectively. κ is a hyper-parameter balancing exploration and exploitation. Specifically, small value of κ indicates that $\mu(\{a_j\}_j, \{r_j\}_j)$ is more important and large value of κ indicates that $\sigma(\{a_j\}_j, \{r_j\}_j)$ is more important. Here, κ is fixed to 2.576, which is the default value in scikit-learn [39].

F. Discussion of limitations

Although InsRect is a flexible and effective framework in features rectification, its achieved improvement relies on the adopted auxiliary OOD datasets. Unreliable auxiliary OOD datasets, e.g., wild OOD data [23], may misguide the removal of misleading information, detrimental for the OOD performance of InsRect. In the future, we will explore how to quantify the quality of auxiliary OOD data and study new techniques that are free from its different choices.

G. Scaling Rule

Following the previous work [8], we also scale the embedding features after rectification and only acts on a portion of elements with large values, which can improve the stability of the rectification results. Denote the original embedding features after InsRect as $\bar{\mathbf{e}}(\mathbf{x})$, the scaling rule can be written as

$$\mathtt{Scale}_i(\bar{\mathbf{e}}(\mathbf{x}_k), p) = \begin{cases} \bar{\mathbf{e}}_i(\mathbf{x}_k)\,\mathrm{R}(\bar{\mathbf{e}}(\mathbf{x}_k), p) & \bar{\mathbf{e}}_i(\mathbf{x}_k) \geq \rho(\bar{\mathbf{e}}(\mathbf{x}_k), p) \\ \bar{\mathbf{e}}_i(\mathbf{x}_k) & \bar{\mathbf{e}}_i(\mathbf{x}_k) < \rho(\bar{\mathbf{e}}(\mathbf{x}_k), p) \end{cases} \tag{19}$$

where $\rho(\bar{\mathbf{e}}(\mathbf{x}_k), p)$ is the p-th percentile of $\bar{\mathbf{e}}(\mathbf{x}_k)$ and $\mathrm{R}(\bar{\mathbf{e}}(\mathbf{x}_k), p)$ is the ratio of the sum of all elements in $\bar{\mathbf{e}}(\mathbf{x}_k)$ to the sum of elements in $\bar{\mathbf{e}}(\mathbf{x}_k)$ greater than $\rho(\bar{\mathbf{e}}(\mathbf{x}_k), p)$. Thereafter, we use

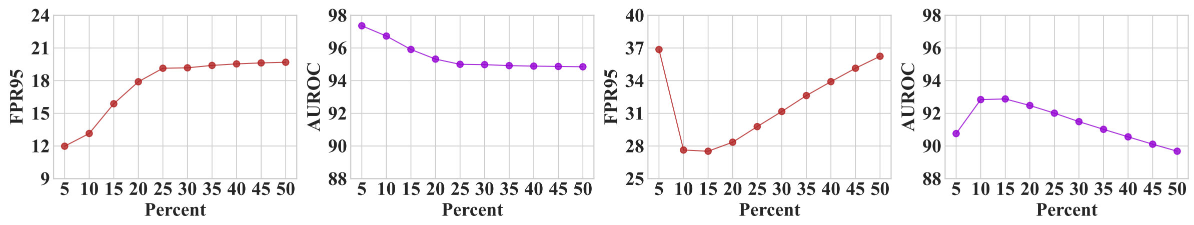
$$\bar{\mathbf{e}}(\mathbf{x}_k) \leftarrow \mathbf{Scale}(\bar{\mathbf{e}}(\mathbf{x}_k), p) \tag{20}$$

as the improved embedding features. Note that we introduce an extra hyper-parameter p when employing the scaling rule, of which the tuning strategy is discussed in the following.

Table 13 summaries the adopted p on both the CIFAR and ImageNet-1K benchmarks, which are selected based on validation set. We list the results across different p on the validation set in Figure 4. Moreover, using auxiliary OOD data to choose p can also lead to promising real performance, where we illustrate the results across different p on auxiliary OOD data in Figure 5. As we can see, the selected p, based on auxiliary OOD data, pretty aligns with the choice based on validation set, demonstrating that the choice of p is insensitive to different OOD datasets.

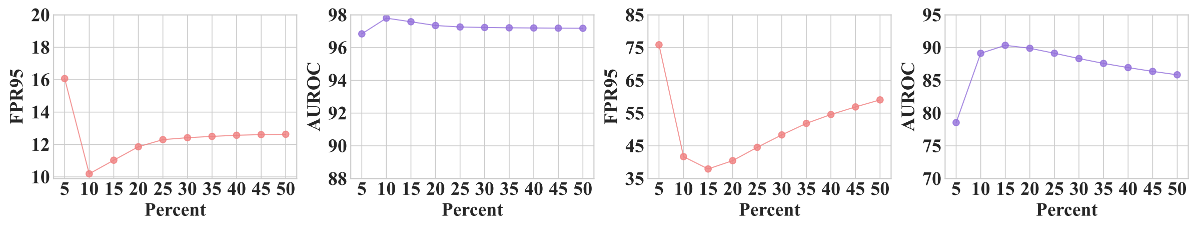
Table 13. The adopted p on the CIFAR and ImageNet-1K benchmark.

Backbones/ID datasets	CIFAR-10	CIFAR-100	ImageNet-1K
ResNet-50	50	35	10
MobileNetV2	/	/	15
WRN-40(50)-2	25	35	20
DenseNet-101(161)	5	15	3



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

Figure 4. The performance of OOD detection on validation set with different hyper-parameter p varying from 5 to 50. As we can see, the optimal hyper-parameter p are 5 and 15 for CIFAR-10 and CIFAR-100, respectively.



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

Figure 5. The performance of OOD detection on auxiliary OOD data with different hyper-parameter p varying from 5 to 50. As we can see, the optimal hyper-parameter p are 10 and 15 for CIFAR-10 and CIFAR-100, respectively.

Table 14. The detailed results (include six common OOD datasets on the CIFAR benchmark) of InsRect adopting different forms of score (include MSP, MaxLogit, and Energy) on CIFAR-10.

CIFAR-10	Textures		Places365		LSUN-Crop		LSUN-Resize		iSUN		SVHN		Average	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
MSP	100.00	74.89	100.00	69.84	100.00	82.82	100.00	76.85	100.00	77.66	100.00	86.50	100.00	78.09
MaxLogit	**20.62**	**95.71**	27.59	93.57	3.76	99.28	9.57	**98.04**	8.20	**98.30**	**4.22**	**99.14**	12.33	97.34
Energy	20.68	95.63	**26.27**	**93.80**	**3.53**	**99.31**	**9.19**	98.02	**7.93**	98.28	4.27	99.10	**11.98**	**97.36**

Table 15. The detailed results (include six common OOD datasets on the CIFAR benchmark) of InsRect adopting different forms of score (include MSP, MaxLogit, and Energy) on CIFAR-100.

CIFAR-100	Textures		Places365		LSUN-Crop		LSUN-Resize		iSUN		SVHN		Average	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
MSP	100.00	73.88	100.00	69.48	40.16	81.27	40.16	80.89	40.16	81.51	100.00	77.55	70.08	77.43
MaxLogit	**54.88**	87.39	**54.33**	84.00	15.52	96.51	10.26	97.37	9.51	97.69	21.87	93.49	27.73	92.74
Energy	55.32	**87.48**	54.51	**84.19**	**14.93**	**96.58**	**9.82**	**97.55**	**9.24**	**97.82**	**21.28**	**93.64**	**27.52**	**92.88**

H. Different Scoring Functions

By default, InsRect uses the form of energy score to indicate the confidence of OOD. However, our method is general in combining with other scoring forms, where we consider two examples of MSP and MaxLogit in Table 14-15. As we can see, InsRect boosts their OOD performance compared with the results in Table 10- 11, revealing that features rectification also is important for many other scoring strategies.

Table 16. The detailed results (include six common OOD datasets on the CIFAR benchmark) of mapping function-based rectification strategy with different forms of score on CIFAR-10.

CIFAR-10	Textures		Places365		LSUN-Crop		LSUN-Resize		iSUN		SVHN		Average	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
MF	46.74	89.93	**26.02**	93.77	6.01	98.68	**7.69**	**98.24**	8.62	98.13	18.51	93.97	18.93	95.45
InsRect	**20.68**	**95.63**	26.27	**93.80**	**3.53**	**99.31**	9.19	98.02	**7.93**	**98.28**	**4.27**	**99.10**	**11.98**	**97.36**

Table 17. The detailed results (include six common OOD datasets on the CIFAR benchmark) of mapping function-based rectification strategy with different forms of score on CIFAR-100.

CIFAR-100	Textures		Places365		LSUN-Crop		LSUN-Resize		iSUN		SVHN		Average	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
MF	82.22	79.60	77.53	71.55	31.70	92.93	**4.77**	**98.99**	**5.70**	**98.84**	87.04	57.70	48.16	83.27
InsRect	**55.32**	**87.48**	**54.51**	**84.19**	**14.93**	**96.58**	9.82	97.55	9.24	97.82	**21.28**	**93.64**	**27.52**	**92.88**

Table 18. The detailed results (include six common OOD datasets on the CIFAR benchmark) of tuning the clamping thresholds manually with different number of common components varying from 1 to 5 on CIFAR-10.

CIFAR-100	Textures		Places365		LSUN-Crop		LSUN-Resize		iSUN		SVHN		Average	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
1	24.14	95.02	25.92	94.03	3.82	99.23	9.13	98.07	8.04	98.33	4.20	99.12	12.54	97.30
2	23.34	95.01	27.35	93.55	4.02	99.20	10.42	97.81	9.01	98.10	3.72	99.22	12.98	97.15
3	22.04	95.30	25.92	93.84	3.82	99.24	9.52	97.93	8.20	98.21	4.13	99.16	12.27	97.28
4	**20.80**	**95.41**	27.06	93.60	4.02	99.23	9.82	97.94	8.42	98.23	**3.37**	**99.27**	12.25	97.28
5	23.80	95.08	**25.77**	**94.11**	**3.14**	**99.36**	**8.35**	**98.23**	**7.41**	**98.44**	4.25	99.09	**12.12**	**97.38**

I. Linear Mapping

Besides clamping common components, using a linear mapping function may also facilitate features rectification, seemingly easier in implementation than our common component-based rectification. However, we suggest that such a mapping function-based (MF) rectification strategy is not preferred in OOD detection, since it introduces too many parameters that are hard for tuning and easy to overfit. To verify our claim, building upon the embedding features, we adopt an additional fully connected layer, whose outputs should be of the same size as the embedding feature of each data point. Then, with the gradient-based parameter optimization and outlier exposure-based learning objective, we find a linear mapping that can facilitate OOD detection. Tables 16-17 demonstrate the detection performance for such a MF rectification strategy. Here, the average performance of MF is much lower than that of InsRect, demonstrating the superiority of our design choice.

J. Manual Tuning

As mentioned in Section 1, By default, we use BO for parameter tuning, while we can also manually tune the thresholds when the number of common components is small (e.g., $1 \sim 5$). We list the results across different choices of J between 1 and 5 in Tables 18-19, guided by the validation performance. Unfortunately, the results are not as impressive as that of the BO tuned parameters, especially for harder tasks as CIFAR-100. Therefore, by default, we suggest using BO for parameter tuning, which are more efficient and effective than manual tuning.

Table 19. The detailed results (include six common OOD datasets on the CIFAR benchmark) of tuning the clamping thresholds manually with different number of common components varying from 1 to 5 on CIFAR-100.

CIFAR-100	Textures		Places365		LSUN-Crop		LSUN-Resize		iSUN		SVHN		Average	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
1	68.49	83.70	59.85	79.09	**28.27**	**93.53**	50.99	81.76	48.97	83.22	27.63	91.63	47.37	85.49
2	**61.17**	**85.45**	61.00	77.47	28.70	93.23	44.14	84.47	41.71	**85.83**	**27.14**	**92.34**	**43.98**	**86.47**
3	79.54	77.58	**55.13**	**80.58**	31.88	91.62	48.05	82.29	46.48	83.27	36.06	86.58	49.52	83.65
4	66.44	81.62	72.73	70.52	35.90	90.30	43.26	83.45	42.97	83.61	30.07	90.73	48.56	83.37
5	64.47	81.36	76.01	67.44	29.39	92.10	**38.21**	**85.63**	**39.78**	85.07	31.91	89.96	46.63	83.59

K. Error Part

By default, we preserve the error part ϵ without specified operations upon them. In Tables 20-21, we further conduct experiment on whether to discard the error part (w/o ϵ). As we can see, discarding the error part will decrease the OOD

Table 20. The detailed results (include six common OOD datasets on the CIFAR benchmark) of whether to discard error part on CIFAR-10.

CIFAR-10	Textures		Places365		LSUN-Crop		LSUN-Resize		iSUN		SVHN		Average	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
w/o ϵ	83.75	83.01	51.41	88.62	7.01	98.58	27.68	93.91	26.39	94.30	48.34	90.40	40.76	91.47
InsRect	**20.68**	**95.63**	**26.27**	**93.80**	**3.53**	**99.31**	**9.19**	**98.02**	**7.93**	**98.28**	**4.27**	**99.10**	**11.98**	**97.36**

Table 21. The detailed results (include six common OOD datasets on the CIFAR benchmark) of whether to discard error part on CIFAR-100.

CIFAR-100	Textures		Places365		LSUN-Crop		LSUN-Resize		iSUN		SVHN		Average	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
w/o ϵ	70.52	82.34	61.69	81.13	21.88	94.78	20.75	94.14	18.13	95.11	30.28	89.81	37.21	89.55
InsRect	**55.32**	**87.48**	**54.51**	**84.19**	**14.93**	**96.58**	**9.82**	**97.55**	**9.24**	**97.82**	**21.28**	**93.64**	**27.52**	**92.88**

Table 22. The detailed results (include six common OOD datasets on the CIFAR benchmark) of adopting global and local thresholds on CIFAR-10.

CIFAR-10	Textures		Places365		LSUN-Crop		LSUN-Resize		iSUN		SVHN		Average	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
Local	24.16	94.88	**25.44**	**94.07**	3.54	99.26	**8.91**	**98.02**	8.05	98.27	**4.02**	**99.12**	12.35	97.27
Global	**20.68**	**95.63**	26.27	93.80	**3.53**	**99.31**	9.19	**98.02**	**7.93**	**98.28**	4.27	99.10	**11.98**	**97.36**

Table 23. The detailed results (include six common OOD datasets on the CIFAR benchmark) of adopting global and local thresholds on CIFAR-100.

CIFAR-100	Textures		Places365		LSUN-Crop		LSUN-Resize		iSUN		SVHN		Average	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
Local	**53.92**	87.22	59.13	81.93	16.71	96.50	13.63	96.52	13.53	97.20	**17.67**	**95.06**	29.10	92.47
Global	55.32	**87.48**	**54.51**	**84.19**	**14.93**	**96.58**	**9.82**	**97.55**	**9.24**	**97.82**	21.28	93.64	**27.52**	**92.88**

Table 24. The detailed results (include six common OOD datasets on the CIFAR benchmark) of OE, InsRect and InsRect combined with OE on CIFAR-10.

CIFAR-10	Textures		Places365		LSUN-Crop		LSUN-Resize		iSUN		SVHN		Average	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
OE	48.86	93.11	**19.04**	**95.75**	0.87	**99.76**	**0.02**	**99.99**	**0.03**	**99.99**	7.38	98.63	12.70	97.87
InsRect	**20.68**	**95.63**	26.27	93.80	3.53	99.31	9.19	98.02	7.93	98.28	**4.27**	**99.10**	11.98	97.36
InsRect+OE	40.67	94.04	21.06	95.28	**0.86**	**99.76**	**0.02**	**99.99**	**0.03**	**99.99**	6.08	98.90	**11.45**	**97.99**

performance, indicating that the error part is also important to boost OOD detection.

L. Global vs Local Thresholds

As mentioned in Section 4, we compute each clamping threshold a_j (or r_j) by an associated proportion p_{a_j} (or p_{r_j}), such that $a_j = \rho(\{\mathbf{c}_j\}_j, p_{a_j})$ (or $r_j = \rho(\{\mathbf{c}_j\}_j, p_{r_j})$). Therefore, in Tables 22-23, we further conduct experiment on adopting local thresholds as $a_j = \rho(\mathbf{c}_j, p_{a_j})$ (or $r_j = \rho(\mathbf{c}_j, p_{r_j})$), and the results reveal that using global thresholds, employed in InsRect by default, is better than that of local thresholds.

M. Outlier Exposure

Besides post-hoc OOD detection with conventionally trained models, our InsRect can also improve OOD detection for OE-trained models, where the results are summarized in Tables 24-25. As we can see, not only OE can improve our InsRect, our InsRect can also improve the OOD detection results of OE.

N. Visualization of the Patterns

Tables 6-12 show visualization of the patterns for different methods (and blank control), including ReAct, ASH and InsRect with different methods in generating common components. As we can see, InsRect provides a more flexible strategy for rectification than ReAct or ASH. Besides, InsRect adopting NMF has a more apparent rectification for embedding features compared with InsRect adopting PCA or ICA, which implies that InsRect adopting NMF is more effective for removing

Table 25. The detailed results (include six common OOD datasets on the CIFAR benchmark) of OE, InsRect and InsRect combined with OE on CIFAR-100.

CIFAR-100	Textures		Places365		LSUN-Crop		LSUN-Resize		iSUN		SVHN		Average	
	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
OE	84.81	83.74	**49.37**	**85.50**	**3.21**	**99.27**	**0.00**	**100.00**	**0.01**	**99.99**	26.89	93.79	27.38	93.72
InsRect	55.32	87.48	54.51	84.19	14.93	96.58	9.82	97.55	9.24	97.82	**21.28**	93.64	27.52	92.88
InsRect+OE	**49.39**	**91.10**	51.37	84.62	6.46	98.71	0.04	99.99	0.05	**99.99**	21.88	**95.14**	**21.53**	**94.92**

misleading information (as in our default implementation). Tables 13-19 show visualization of the patterns for different iterations including 0, 20, 50, 100, 200, 500. As we can see, the effects of rectification becomes apparent from 100 iterations and stabilizes after 200 iterations, which is consistent with the results of experiment mentioned in Section 4.2.

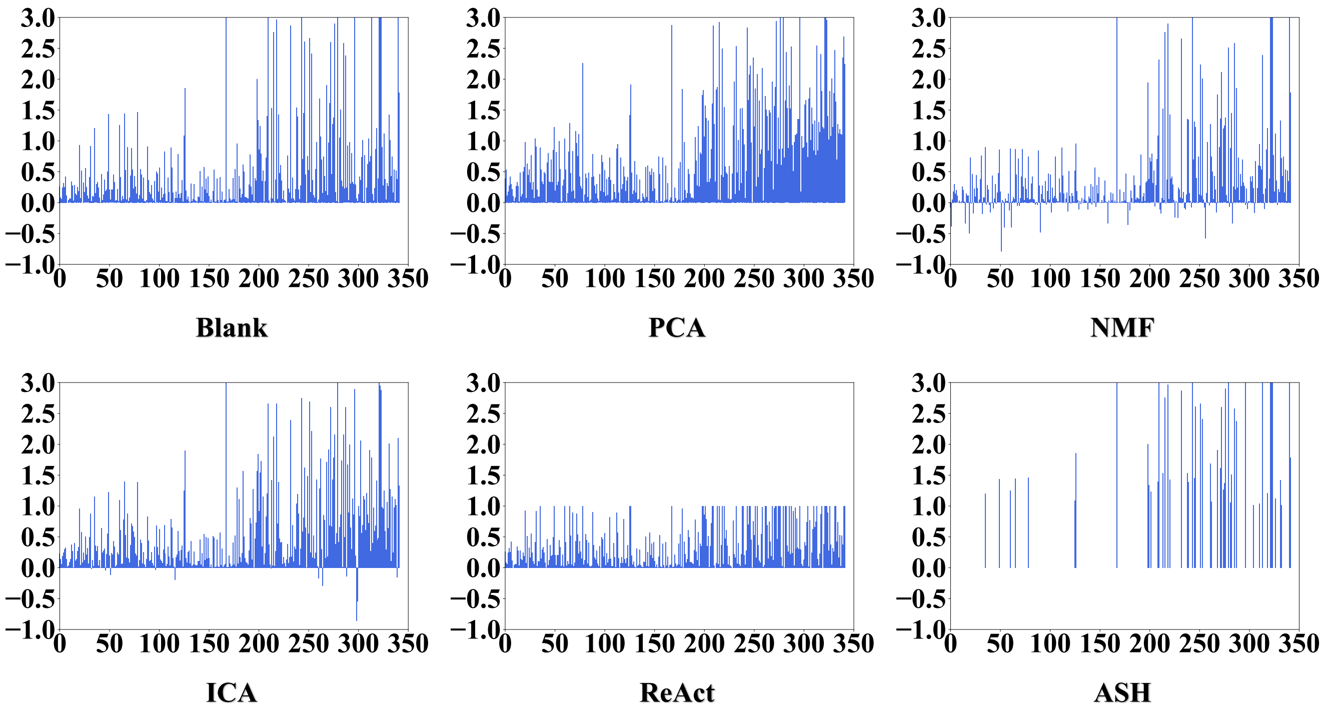


Figure 6. The patterns of CIFAR-100 for different methods (and blank control).

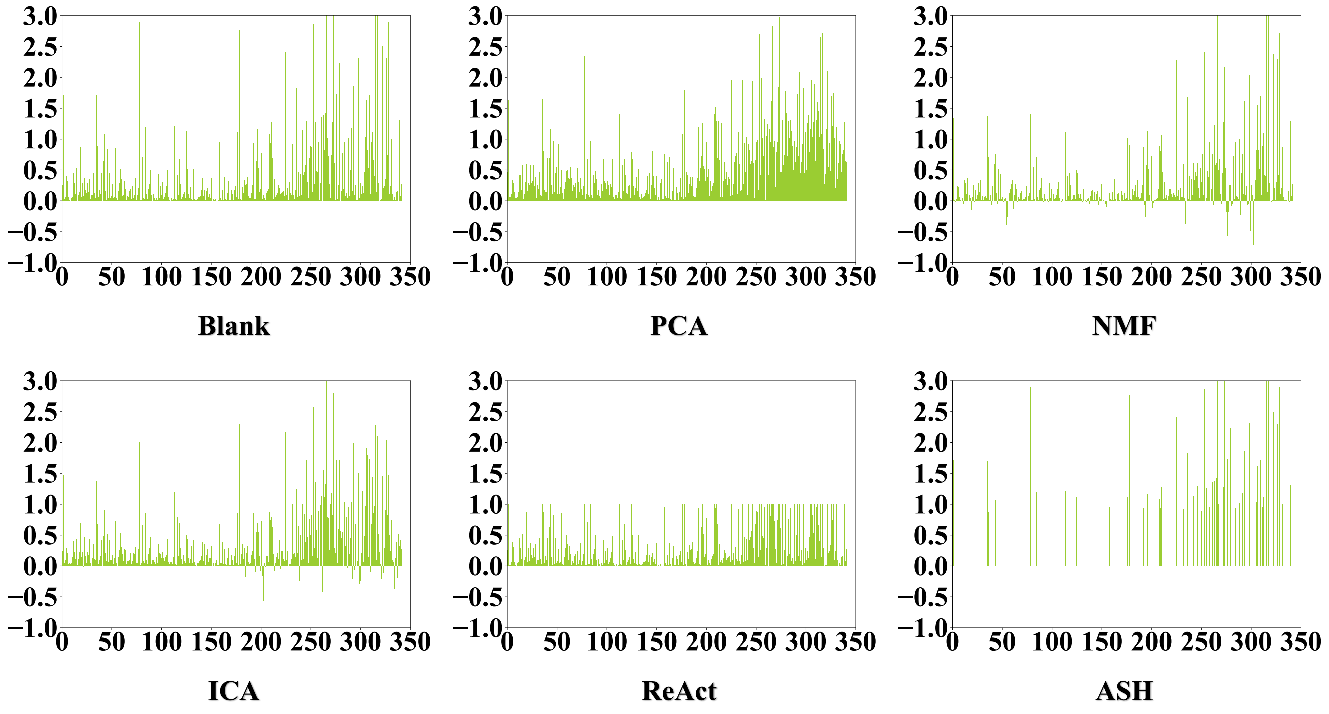


Figure 7. The patterns of Textures for different methods (and blank control).

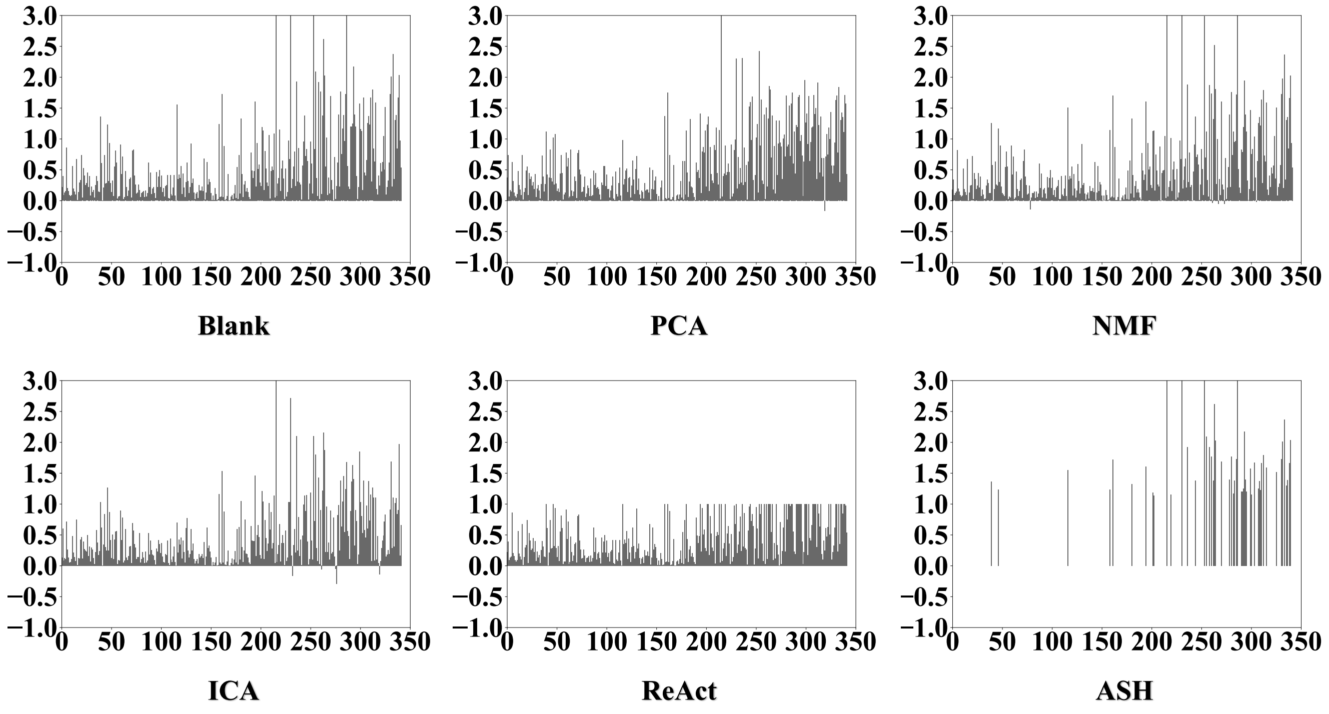


Figure 8. The patterns of Places365 for different methods (and blank control).

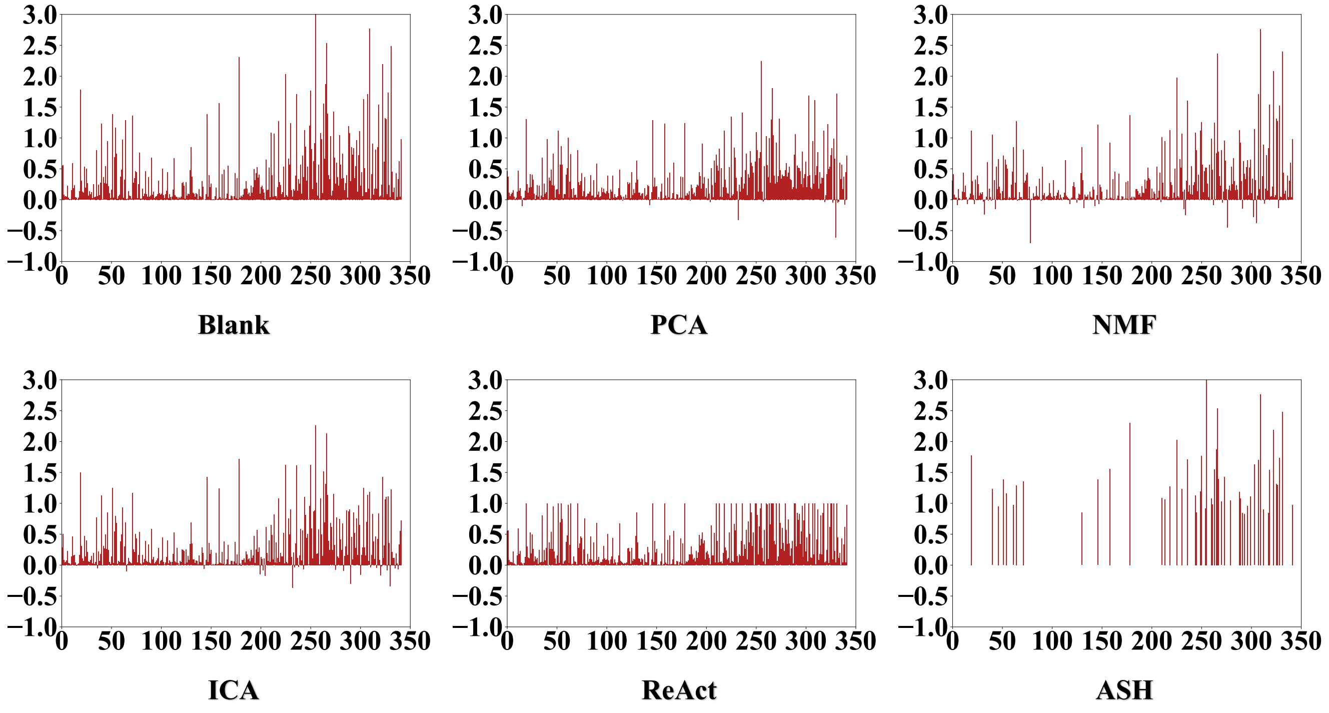


Figure 9. The patterns of LSUN-Crop for different methods (and blank control).

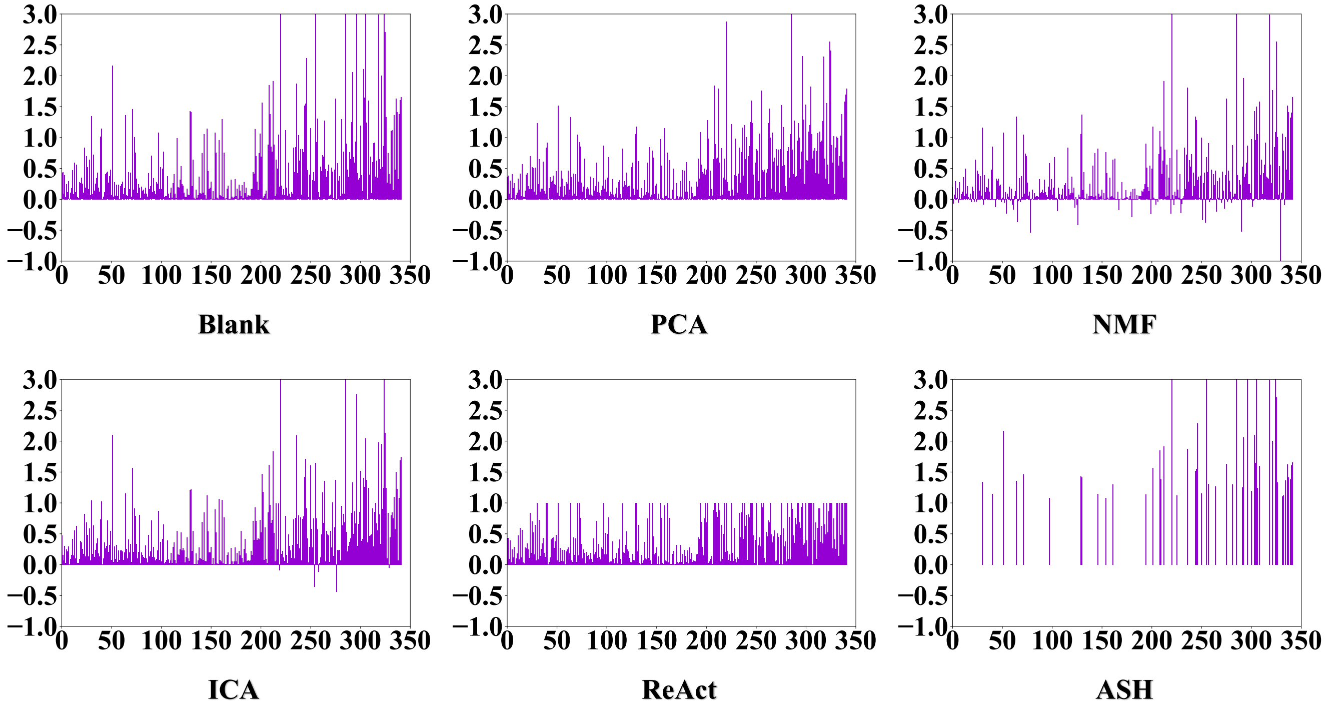


Figure 10. The patterns of LSUN-Resize for different methods (and blank control).

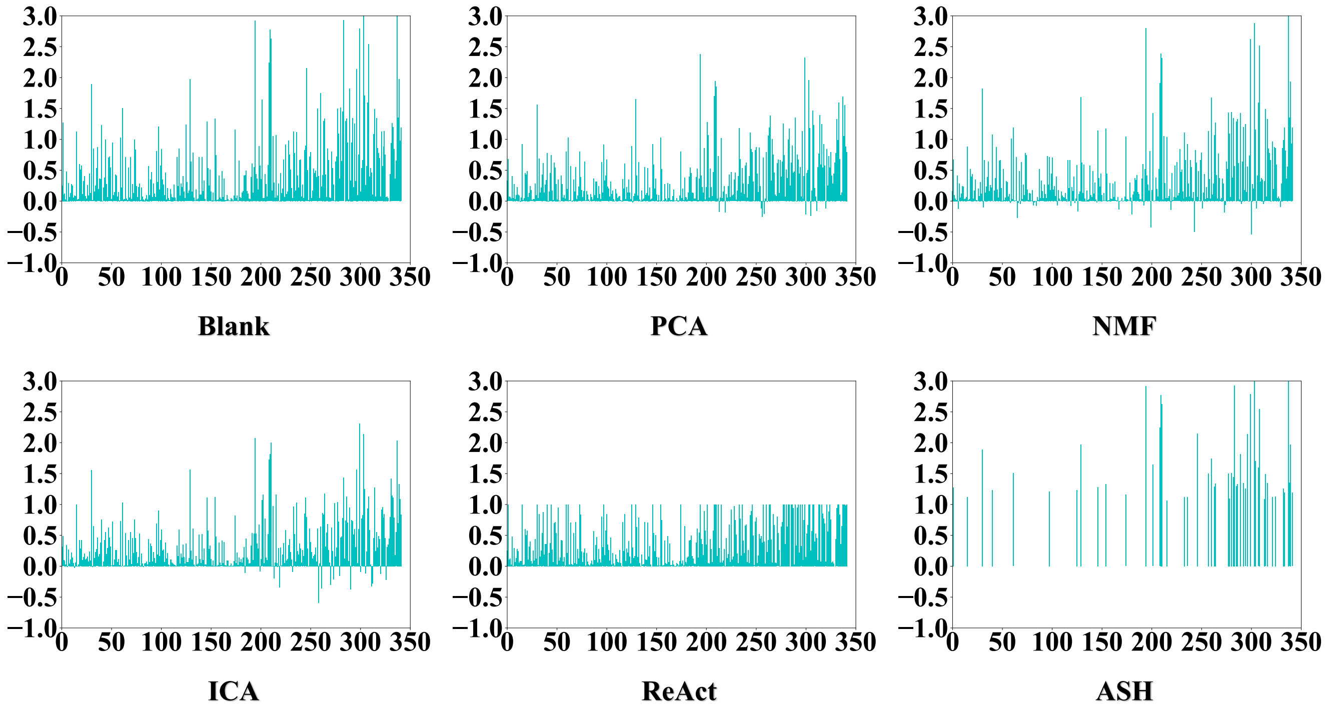


Figure 11. The patterns of iSUN for different methods (and blank control).

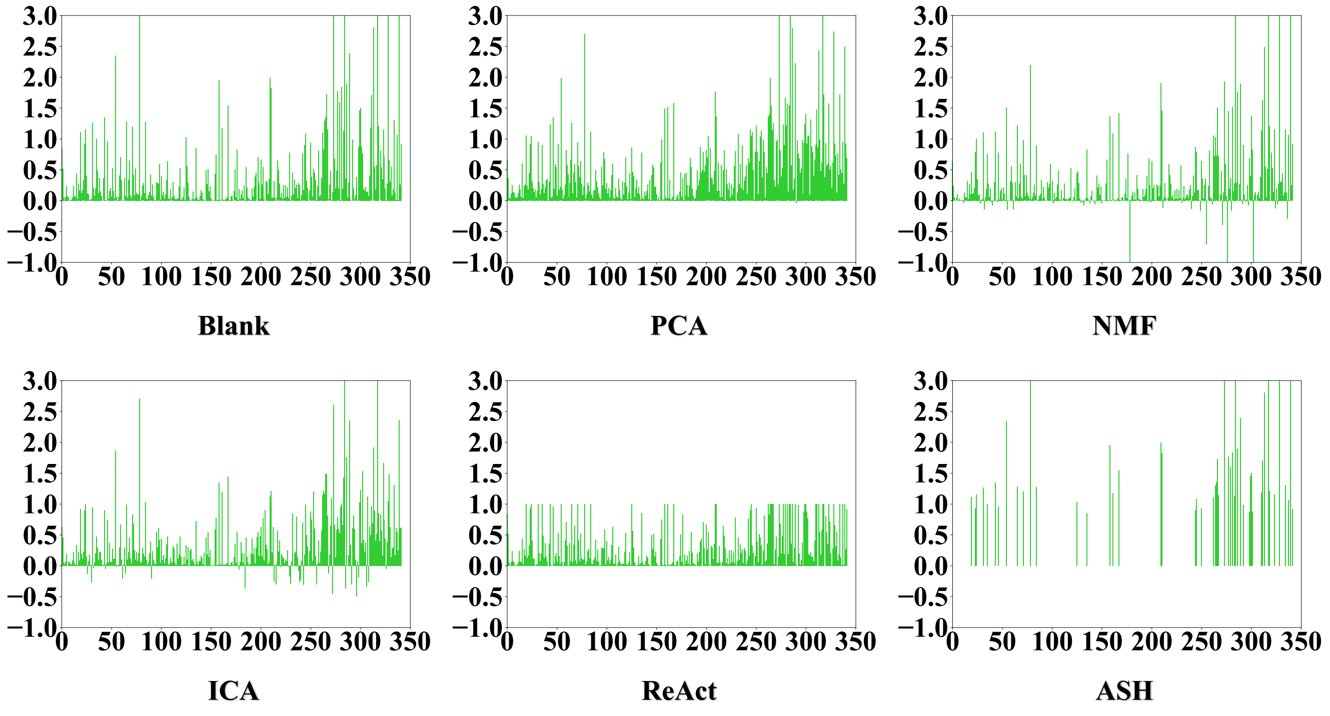


Figure 12. The patterns of SVHN for different methods (and blank control).

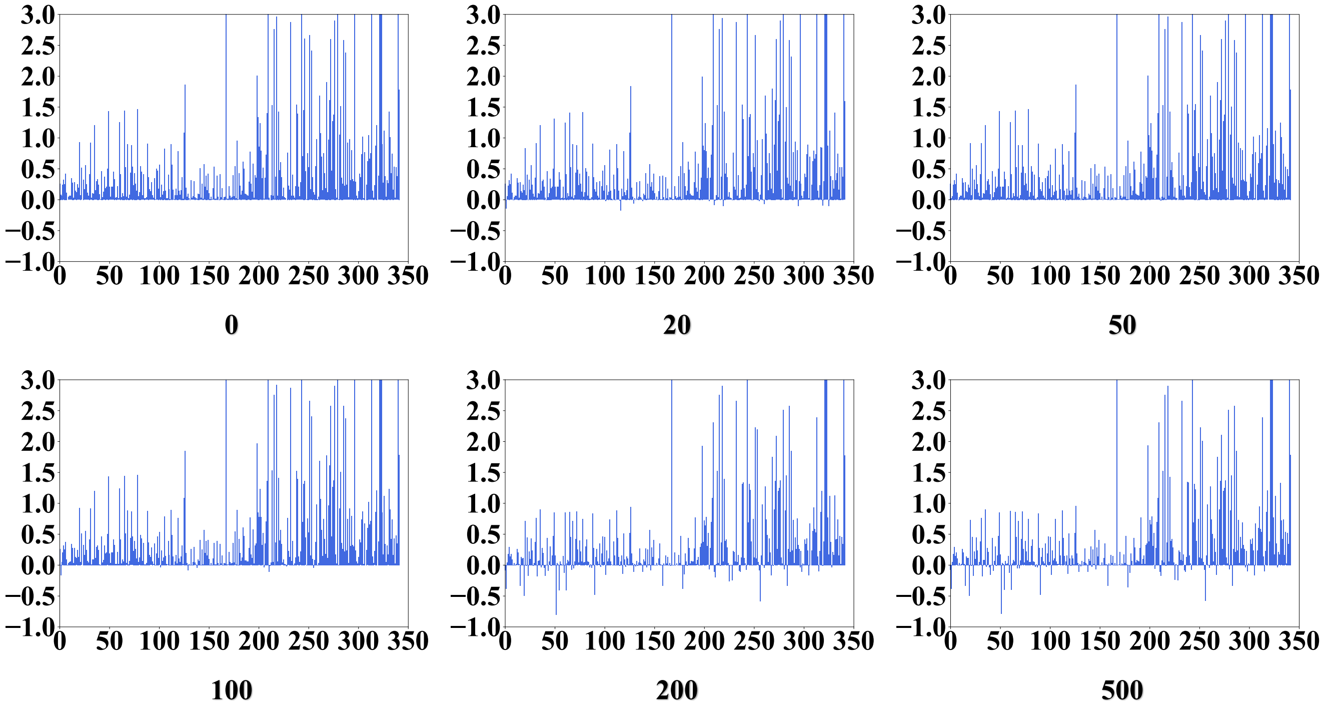


Figure 13. The patterns of CIFAR-100 for different iterations.

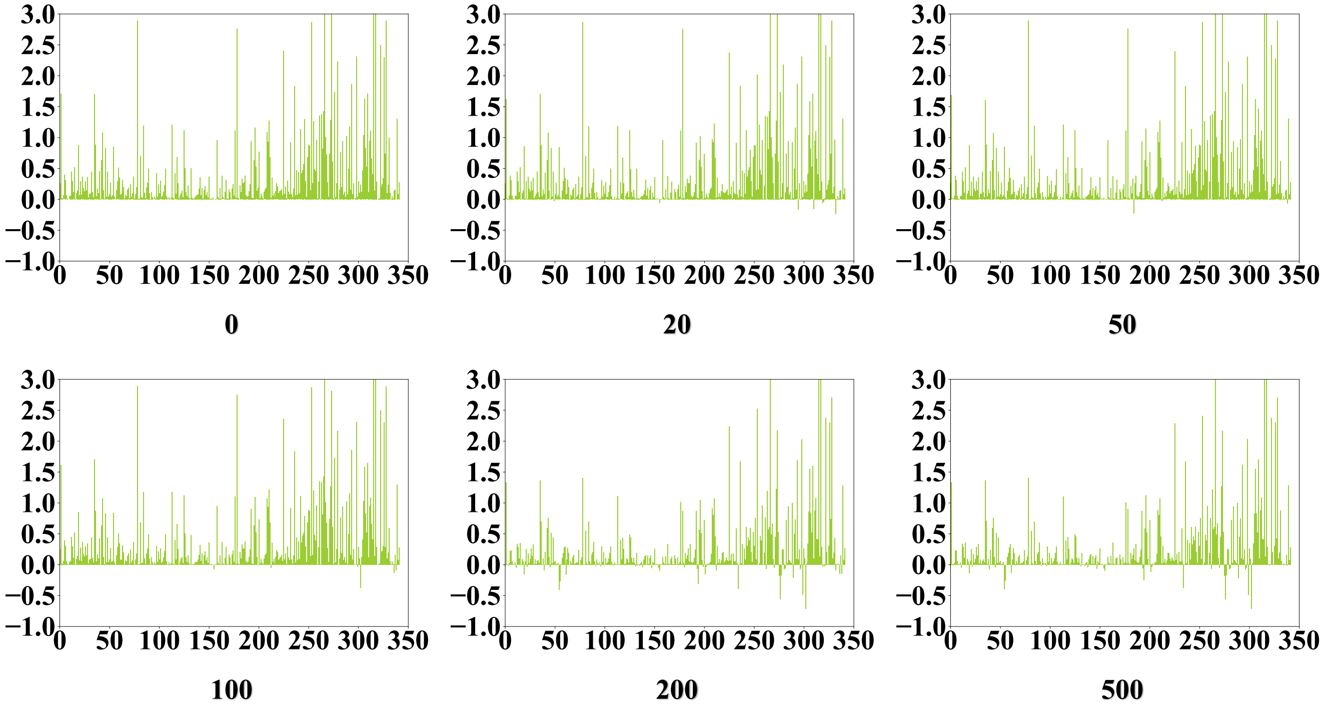


Figure 14. The patterns of Textures for different iterations.

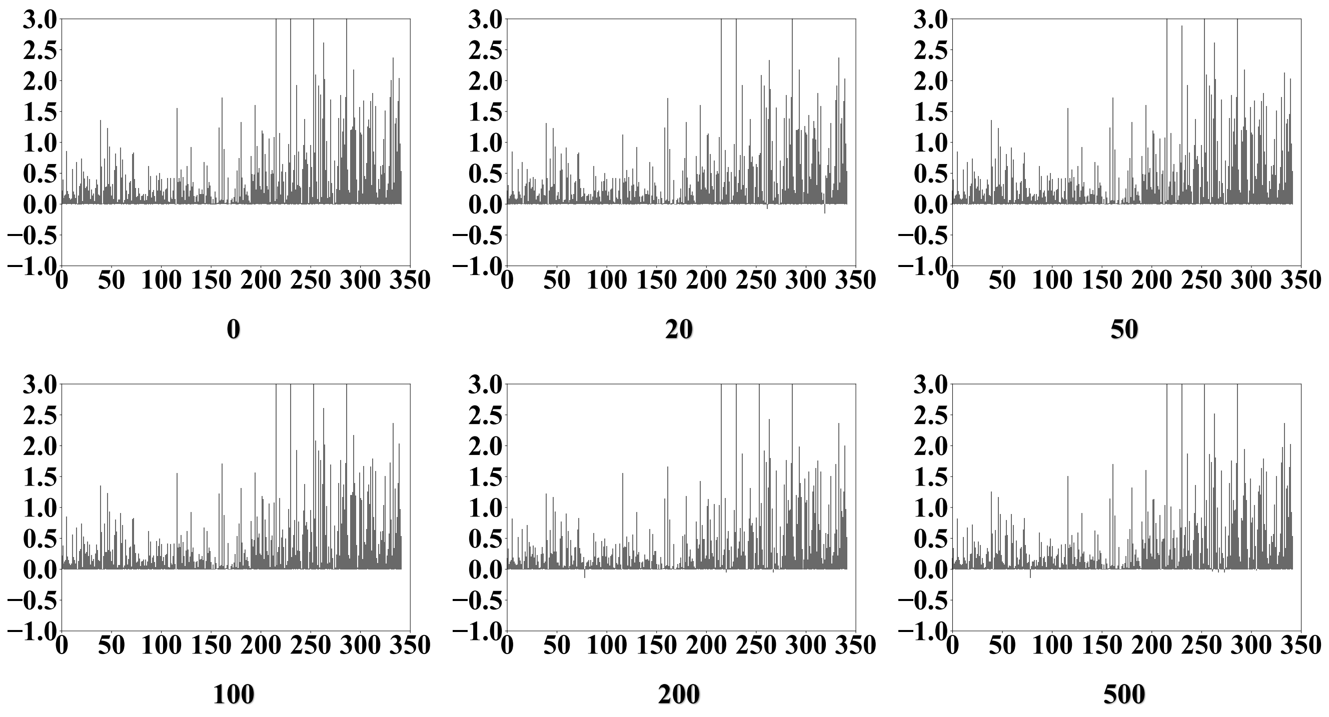


Figure 15. The patterns of Places365 for different iterations.

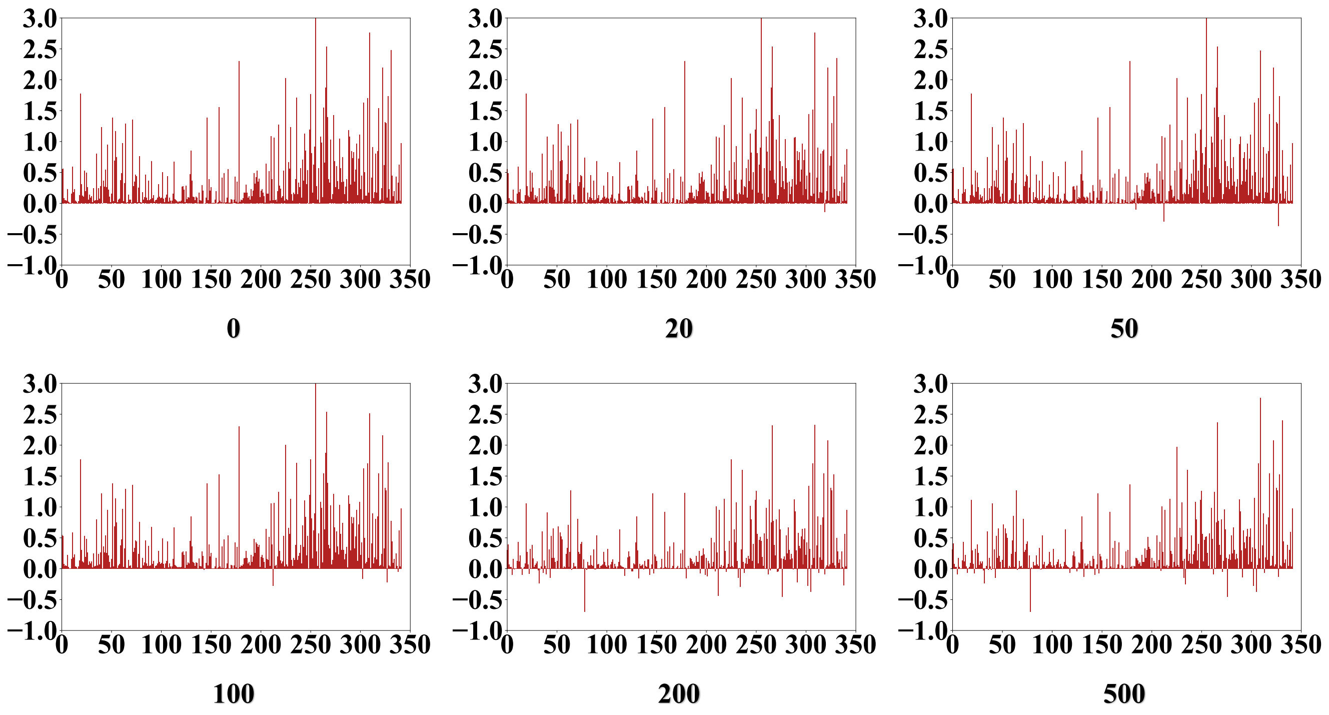


Figure 16. The patterns of LSUN-Crop for different iterations.

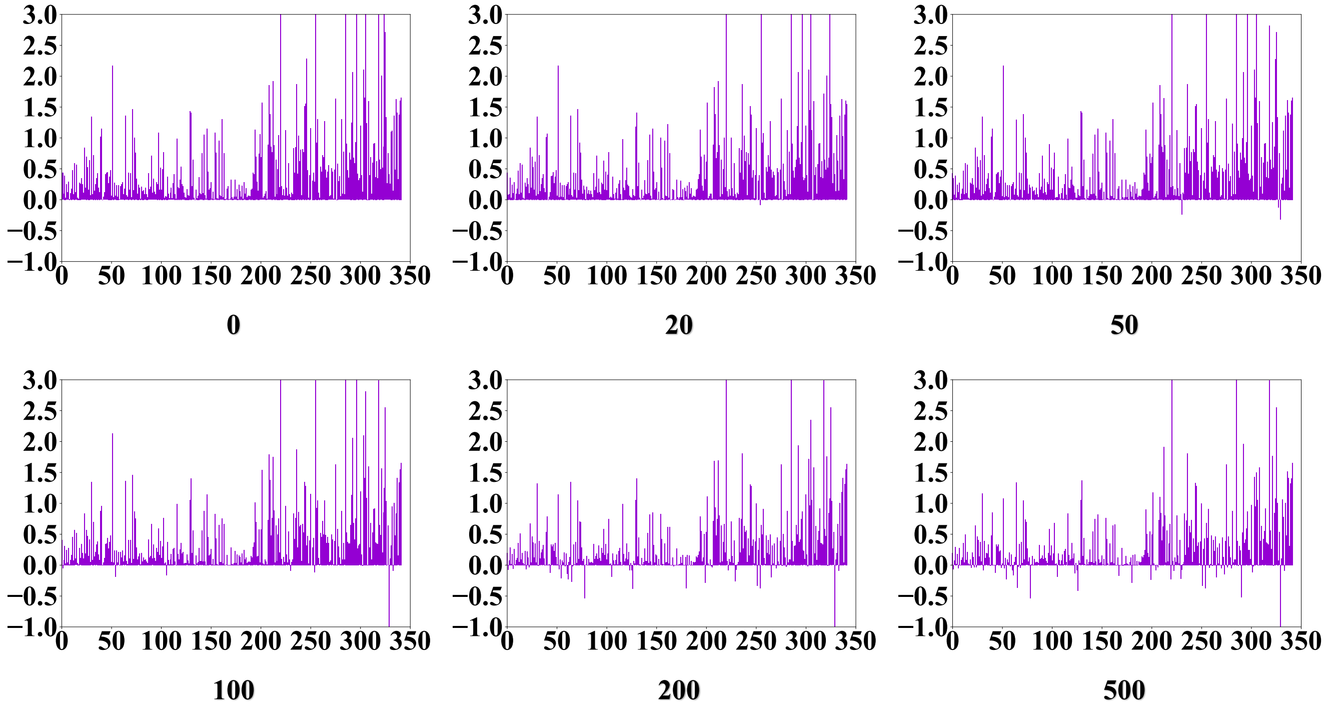


Figure 17. The patterns of LSUN-Resize for different iterations.

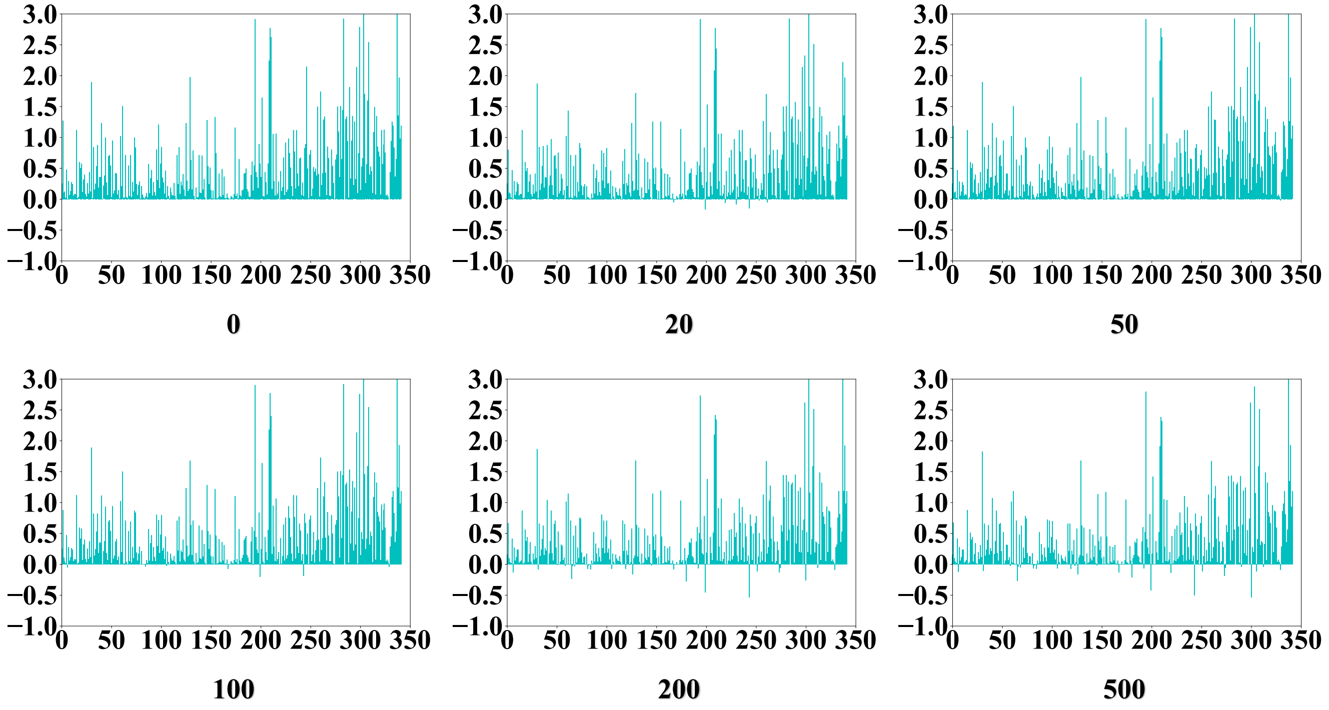


Figure 18. The patterns of iSUN for different iterations.

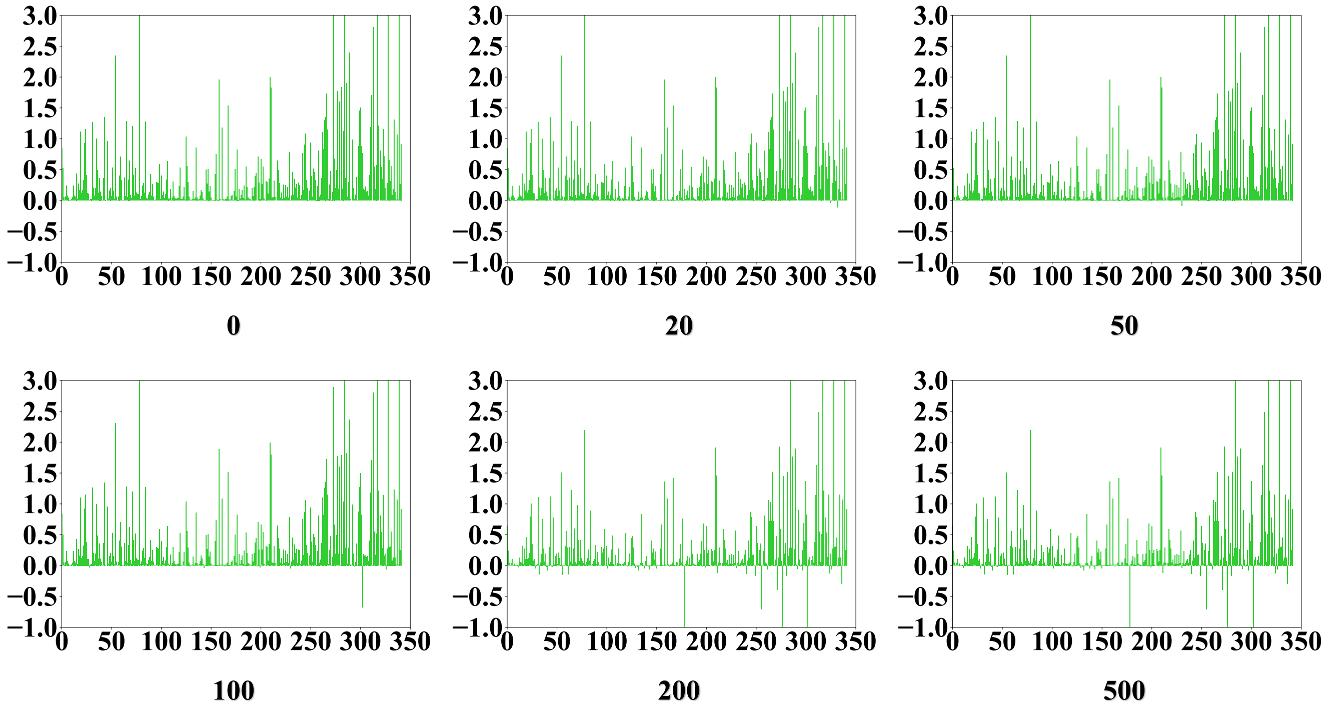


Figure 19. The patterns of SVHN for different iterations.