Boosting Out-of-distribution Detection with Typical Features

Anonymous Author(s)
Affiliation
Address
email

Abstract

Out-of-distribution (OOD) detection is a critical task for ensuring the reliability and safety of deep neural networks in real-world scenarios. Different from most previous OOD detection methods that focus on designing OOD scores or introducing diverse outlier examples to retrain the model, we delve into the obstacle factors in OOD detection from the perspective of typicality and regard the feature's high-probability region of the deep model as the feature's typical set. We propose to rectify the feature into its typical set and calculate the OOD score with the typical features to achieve reliable uncertainty estimation. The feature rectification can be conducted as a plug-and-play module with various OOD scores. We evaluate the superiority of our method on both the commonly used benchmark (CIFAR) and the more challenging high-resolution benchmark with large label space (ImageNet). Notably, our approach outperforms state-of-the-art methods by up to 5.11% in the average FPR95 on the ImageNet benchmark.

1 Introduction

Deep neural networks have been widely applied in various fields. Apart from the success of deep models, predictive uncertainty is essential in safety-critical real-world scenarios such as autonomous driving [1, 2], medical [3], financial [4], etc. When encountering some examples that the deep model has not been exposed to during training, we hope the model raises an alert and hands them over to humans for safe handling. Such a challenge is usually referred to as out-of-distribution (OOD) detection and has gained significant research attention recently [5–8].

Most of the existing research [5–7, 9–12] worked on designing suitable OOD scores for the pre-trained neural network, hoping to assign higher scores to the in-distribution (ID) examples and lower scores to the out-of-distribution (OOD) examples. However, these methods overlook the obstacle factors in OOD detection caused by the model's internal mechanisms. In this paper, we rethink the OOD detection from a perspective of feature typicality. We observed that the distribution of the deep features of the training dataset on different channels is approximately consistent with the Gaussian distribution (See examples in Appendix J). Accordingly, we divide these features into typical features (fall in the high-probability region) and extreme features (fall in the low-probability region). Extreme features rarely appear in training and attract less attention from the classifier than the typical features. We hypothesize the classifier can model the typical features better than the extreme features, and the extreme features may lead to ambiguity and imprecise uncertainty estimation. Given the potential negative impact, properly dealing with these extreme features is a key to improving the performance of OOD detection.

In this paper, we propose to rectify the features into their typical set and then calculate the OOD score with these typical features. In this way, the model conservatively utilizes the typical features to make decisions and alleviates the damage caused by extreme features, which can be beneficial to the

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

OOD scores derived from the pre-trained classifier [5–7, 12]. Then the problem is how to estimate the feature's typical set on different channels since this requires a sufficient number of in-distribution examples and is time-consuming. Luckily, the commonly used operation Batch Normalization can shed light on a shortcut to selecting the feature's typical set and we name our approach **B**atch Normalization **A**ssisted **T**ypical **S**et Estimation (**BATS**). The Batch Normalization layer endeavors to normalize the features of the training dataset to Gaussian distributions, which can be used to estimate the typical set for ID features. Typical features are more common in training, while extreme features are rare, which leads to difficulties for the model to estimate extreme features well. We truncate the deep features with the guidance of the Batch Normalization, rectifying the extreme features to the boundary values of typical sets. We illustrate the distribution of the OOD scores for ID (ImageNet) and OOD (four different datasets) examples in Fig. 1. Rectifying the features into the typical set with our **BATS** contributes to improving the separability between ID and OOD examples.

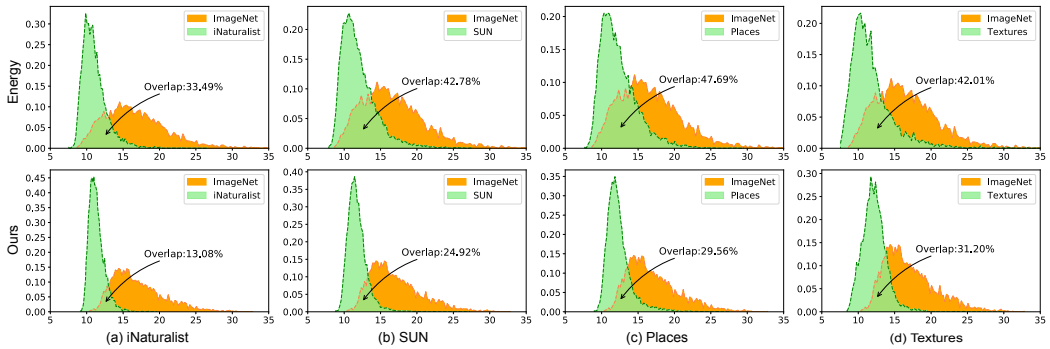


Figure 1: The distribution of the scores for ID (ImageNet) and OOD examples on ResNet-50. We use the energy score [5] as the OOD score. "Energy" means calculating the OOD score with the original features. "Ours" means calculating the OOD score with the typical features.

Theoretically, we analyze the benefit of **BATS** and the bias-variance trade-off influenced by the strength of the hyperparameter. A proper strength of **BATS** contributes to improving the estimation accuracy of the reject region. Empirically, we perform extensive evaluations and establish superior performance on both the large-scale ImageNet benchmark and the commonly used CIFAR benchmarks. **BATS** outperforms the previous best method by a large margin, with up to a 5.11% reduction in the false positive rate (FPR95) and a 1.43% improvement in AUROC. Moreover, **BATS** can also slightly improve the test accuracy and robustness of the pre-trained models.

The main contributions of our paper are summarized as follows:

- We provide novel insights into OOD detection from the perspective of typicality and propose to rectify the features into the typical set. We design a concise and effective approach to select the feature's typical set named **B**atch Normalization **A**ssisted **T**ypical **S**et Estimation (**BATS**).
- We provide theoretical analysis and empirical ablation on the benefit of **BATS** from the perspective of bias-variance trade-off to improve the understanding of our approach.
- Extensive experiments show that **BATS** establishes a state-of-the-art performance among post-hoc methods on a suite of OOD detection benchmarks. Moreover, **BATS** can boost the performance of various existing OOD scores with typical features.

2 Related work

The literature related to OOD detection can be broadly grouped into the following themes: post-hoc detection methods [5–8, 11–13], confidence enhancement methods [9, 10, 14–19], and density-based methods [20–25]. Post-hoc detection methods focus on improving the OOD uncertainty estimation by utilizing the pre-trained classifiers rather than retraining a model, which is beneficial for adopting OOD detection in real-world scenarios and large-scale settings. MSP [6] observes that the maximum softmax probability of ID examples can be higher than that of the OOD examples and provide a

simple baseline for OOD detection. ODIN [7] introduces a sufficiently large temperature factor and input perturbation to separate the ID and OOD examples. Liu et al. [5] analyze the limitations of softmax function in OOD detection and propose to use energy score as an indicator. The examples with high energy are considered OOD examples, and vice versa. ReAct [8] hypothesizes that the OOD examples can trigger the abnormal activation of the model and propose to clamp the activation value larger than the threshold to improve the detection performance. GradNorm [12] shows that the gradients of the categorical cross-entropy loss can be an effective test statistic for OOD detection. Different from these methods, our **BATS** proposes to calculate the OOD scores with the typical features, which benefits the estimation of the reject region and can improve the detection performance.

3 Preliminaries

3.1 Out-of-distribution detection

In this section, we provide a summary of the out-of-distribution detection from the perspective of hypothesis testing [26–30]. We consider a classification problem with K classes and denote the labels as $\mathcal{Y} = \{1, 2, \ldots, K\}$. Let $\mathcal{X}$ be the input space. Suppose that the in-distribution data $\mathcal{D}_{in} = \{(x_i, y_i)\}_{i=1}^{n}$ is drawn from a joint distribution $P_{X,Y}$ defined over $\mathcal{X} \times \mathcal{Y}$. We denote the marginal distribution of $P_{X,Y}$ for the input variable X by P_0. Given a test input $\mathbf{x} \in \mathcal{X}$, the problem of out-of-distribution detection can be formulated as a single-sample hypothesis testing task:

$$\mathcal{H}_0 : \mathbf{x} \sim P_0, \quad \text{vs.} \quad \mathcal{H}_1 : \mathbf{x} \nsim P_0. \tag{1}$$

Here the null hypothesis $\mathcal{H}_0$ implies that the test input $\mathbf{x}$ is an in-distribution sample. The goal of OOD detection here is to design criteria based on $\mathcal{D}_{in}$ to determine whether $\mathcal{H}_0$ should be rejected. OOD detection tasks need to determine a reject region $\mathcal{R}$ such that for any test input $\mathbf{x} \in \mathcal{X}$, the null hypothesis is rejected if $x \in \mathcal{R}$. Generally, the reject region $\mathcal{R}$ is formulated by a test statistic and a threshold. Let $f : \mathcal{X} \mapsto \mathbb{R}^K$ be a model pre-trained from $\mathcal{D}_{in}$, which is used to predict the class label of an input sample. One can use the model f or a part of f (e.g., feature extractor) to construct a test statistic $T(\mathbf{x}; f)$, where $\mathbf{x}$ is the test input. Then the reject region can be written as $\mathcal{R} = \{\mathbf{x} : T(\mathbf{x}; f) \leq \gamma\}$, where γ is the threshold.

3.2 OOD detection with energy score

For a classifier f and a data point $(\mathbf{x}, y)$, we use $f(\mathbf{x})[k]$ to represent the kth output of the last layer. With reference to [5, 31, 32], the marginal density $p(\boldsymbol{x})$ of the classifier can be expressed as: $p(x) = \frac{\exp(-E(x))}{Z} = \frac{\sum_{k=1}^{K} \exp(f(\mathbf{x})[k])}{Z}$, where Z is the normalizing factor and is independent to $\mathbf{x}$. $E(x)$ represents the energy of x and is modeled by neural network as $E(\mathbf{x}) = -\log \sum_{k=1}^{K} \exp(f(\mathbf{x})[k])$. See Appendix C for details. Considering that Z is a constant and is independent to $\mathbf{x}$, Liu et al. [5] propose an energy score that uses the opposite of the energy $E(\mathbf{x})$ as a test statistic to detect OOD examples. A higher energy score means a higher marginal density $p(\boldsymbol{x})$.

4 Methods

In this paper, we delve into the obstacle factor for the post-hoc OOD detection from the perspective of typicality, which aims to boost the performance of the existing OOD scores and is orthogonal to the methods of designing different OOD scores. Given that the energy score [5] is provably aligned with the density of inputs and performs well, we mainly use the energy score as the OOD score (See Appendix I for other OOD scores).

4.1 Motivation

For a classifier trained on the ID data $f = f_{w,b} \circ g$ where g is a feature extractor mapping input $\mathbf{x}$ to its deep feature $\mathbf{z}$. Let d-dimensional vector $\mathbf{z} = [z_1, ..., z_d]^\top = g(\mathbf{x})$ denote the deep features of $\mathbf{x}$ extracted by g, and z_i indicate the i-th element of $\mathbf{z}$. $f_{\mathbf{w},\mathbf{b}}(\mathbf{z}) = \mathbf{w} \cdot \mathbf{z} + \mathbf{b}$ is a fully connected layer mapping the deep feature $\mathbf{z}$ to output logits. The energy can be expressed as:

$$E(\mathbf{x}) = -\log \sum_{k=1}^{K} \exp(f(\mathbf{x})[k]) = -\log \sum_{k=1}^{K} \exp((\mathbf{w} \cdot \mathbf{z} + \mathbf{b})[k]). \tag{2}$$

The test statistic can be expressed as $T(\mathbf{x}; f) := -E(\mathbf{x}) = \log \sum_{k=1}^{K} \exp((\mathbf{w} \cdot \mathbf{z} + \mathbf{b})[k])$, which depends on the extracted deep features and the mapping operation of the fully connected layer (FC). Assuming that the distribution of the deep features is consistent with the Gaussian distribution (see examples in Appendix J), there are high-probability regions and low-probability regions in deep features. We name the features that fall in high-probability regions as typical features, and the corresponding regions are called feature's typical sets. In contrast, we regard the features that fall in low-probability regions as extreme features. Extreme features are rarely exposed to the training process, which leads to difficulties for the classifier to model these features and unreliable estimations in the inference process. Reducing the influence of extreme features on test statistics can be a key to improving OOD detection performance.

4.2 Batch Normalization Assisted Typical Set Estimation

Instead of designing new OOD scores to detect the abnormality, we provide a novel insight into OOD detection from a perspective of typicality. We propose to rectify the features into the feature's typical set and then use these typical features to calculate the OOD score. Consider a commonly used layer structure in deep convolutional networks:

$$\mathbf{z}' \rightarrow \text{BN}(\mathbf{z}'; \mu, \sigma) \rightarrow \text{ReLU} \rightarrow \mathbf{z}, \tag{3}$$

where $\mathbf{z}'$ is the feature vector extracted from the convolutional layer of g. To identify the typical set of $\mathbf{z}'$ for each channel, we should apply its feature map to a sufficient number of ID examples and further calculate the empirical distribution of $\mathbf{z}'$ over the ID examples. If the number of features is large, the inference procedure is time-consuming. *Here we propose a simple and effective post hoc approach that leverages the information stored in f to infer the typical set without estimating the distribution of* $\mathbf{z}'$. Suppose the pre-trained deep neural network uses batch normalization (BN). We denote the BN unit in f as:

$$\text{BN}(\mathbf{z}'; \mu, \sigma) = \sigma \frac{\mathbf{z}' - \mathbb{E}(\mathbf{z}')}{\text{Std}(\mathbf{z}')} + \mu, \tag{4}$$

where μ, σ are two learnable parameters. After the pre-training, all the four parameters μ, σ, $\mathbb{E}(\mathbf{z}')$, $\text{Std}(\mathbf{z}')$ are known and stored in the weights of f.[1] The Batch Normalization normalizes features of the training dataset to a distribution with a mean of μ and standard deviation of σ, which means that the features fall in the interval $[\mu - \lambda * \sigma, \mu + \lambda * \sigma]$ appear more frequently in training than the features in the complement of this interval. The parameter λ controls the range of the interval. Thus we use the information in the Batch Normalization to identify the in-distribution feature's typical set and rectify the features into the typical set before calculating the OOD score. The uncertainty estimated with the typical features can be more reliable.

In practice, we propose a truncated activation scheme to bound the output features of the BN unit. First, we introduce the truncated BN unit by:

$$\text{TrBN}(\mathbf{z}'; \mu, \sigma, \lambda) = \begin{cases} \mu + \lambda\sigma, & \text{if} \quad \mathbf{z}' - \mu \geq \lambda\sigma; \\ \text{BN}(\mathbf{z}'; \mu, \sigma), & \text{if} \quad -\lambda\sigma < \mathbf{z}' - \mu < \lambda\sigma; \\ \mu - \lambda\sigma, & \text{if} \quad \mathbf{z}' - \mu \leq -\lambda\sigma, \end{cases} \tag{5}$$

where λ is a tuning parameter. We replace the BN unit in the layer structure (Eq.(3)) with the TrBN unit and write the rectified final features as $\bar{\mathbf{z}}$ and the new classifier as $\bar{f}$. Then test statistic with the energy score can be expressed as:

$$T(\mathbf{x}; \bar{f}) = \log \sum_{k=1}^{K} \exp((\mathbf{w} \cdot \bar{\mathbf{z}} + \mathbf{b})[k]) \tag{6}$$

and take the reject region by $\mathcal{R} = \{\mathbf{x} : T(\mathbf{x}; \bar{f}) \leq \gamma\}$. We name our approach **B**atch Normalization **A**ssisted **T**ypical **S**et Estimation (BATS). In comparison to the standard BN, the outputs of TrBN are concentrated toward the feature's typical set of ID data. This makes an ID example less susceptible to being mistakenly detected as an OOD example and buffers the negative impact of the extreme features. Fig. 1 compares the distribution of the OOD scores from the original energy score and the energy score with typical features.

[1] In general, $\mathbb{E}(\mathbf{z}')$ and $\text{Std}(\mathbf{z}')$ are estimated on a mini-batch of the training data. Finally, the pre-trained model outputs moving average estimators at each iteration.

4.3 Theoretical analysis

The truncation threshold λ is a key hyperparameter. Our method reduces the variance of $\mathbf{z}'$ and also introduces a bias term since it changes the distribution of $\text{BN}(\mathbf{z}'; \mu, \sigma)$. The variance reduction means that our method is robust to the rare ID examples, while the introduced bias can lead to degradation of the model performance. In this section, we assume $\mathbf{z}'$ follows a normal distribution and analyze the bias-variance trade-off in our method.

4.3.1 Understanding the benefits of BATS from the perspective of variance reduction

The variance reduction happens at the BN step. The variance of $\text{BN}(\mathbf{z}'; \mu, \sigma)$ is σ^2 since the distribution of the ID features $\mathbf{z}'$ is rescaled to $N(\mu, \sigma^2)$. While the TrBN unit truncates the extreme values and the variance of $\text{TrBN}(\mathbf{z}'; \mu, \sigma, \lambda)$ becomes:

$$\sigma^2 C(\lambda) := \sigma^2 \Big(\text{erf}(\frac{\lambda}{\sqrt{2}}) - \frac{\sqrt{2}}{\sqrt{\pi}} \lambda \exp(-\frac{\lambda^2}{2}) + \lambda^2 (1 - \text{erf}(\frac{\lambda}{\sqrt{2}})) \Big), \tag{7}$$

where $\text{erf}(x) = (2/\sqrt{\pi}) \int_0^x \exp(-t^2) dt$ is the Gauss error function. The value of $C(\lambda)$ represents the degree of variance reduction. In Eq.(7), $C(0) = 0$, $dC(\lambda)/d\lambda > 0$, and $C(\lambda) \to 1$ as $\lambda \to +\infty$. Therefore, $C(\lambda)$ is a monotonically increasing function and $0 \leq C(\lambda) < 1$ for $0 \leq \lambda < +\infty$. In summary, the smaller λ, the smaller the variance. See Appendix D for the proof.

OOD detection is a single-sample hypothesis testing problem (in Eq.(1)), and the in-distribution P_0 is unknown. So the reject region is determined by the empirical distribution of the test statistic $T(\mathbf{x}; f)$ over the ID data. The extreme features increase the uncertainty and lead to more unusual values of $T(\mathbf{x}; f)$. This implies that the reject region may be underestimated due to the heavy tail property of $T(\mathbf{x}; f)$. Our **BATS** aids this problem by reducing the variance of the deep features, which contributes to constraining the uncertainty of f and $T(\mathbf{x}; f)$ and improving the estimation accuracy of the reject region.

4.3.2 The bias introduced by BATS

BATS rectifies the features into the typical set, which reduces the variance of the deep features. However, this operation can also introduce a bias term, which can reflect the change in the distribution of the features. A large bias can damage the performance of the model. The distribution of the output feature $\mathbf{z} = \text{ReLU}(\text{BN}(\mathbf{z}'; \mu, \sigma, \lambda))$ is a one-side rectified normal distribution over $[0, +\infty)$. The expectation of $\mathbf{z}$ is:

$$\mathbb{E}(\mathbf{z}) = \mu + \sigma \Big(\frac{1}{\sqrt{2\pi}} \big(\exp(-\frac{\mu^2}{2\sigma^2}) \big) - \frac{\mu}{2\sigma} (1 + \text{erf}(-\frac{\mu}{\sqrt{2}\sigma})) \Big). \tag{8}$$

For our method, the distribution of the output feature $\bar{\mathbf{z}} = \text{ReLU}(\text{TrBN}(\mathbf{z}; \mu, \sigma, \lambda))$ is a two-sided rectified normal distribution over $[0, \mu + \lambda\sigma]$ and the expectation of $\bar{\mathbf{z}}$ is:

$$\mathbb{E}(\bar{\mathbf{z}}) = \mu + \sigma \Big(\frac{1}{\sqrt{2\pi}} \big(\exp(-\frac{\mu^2}{2\sigma^2}) - \exp(-\frac{\lambda^2}{2}) \big) - \frac{\mu}{2\sigma} (1 + \text{erf}(-\frac{\mu}{\sqrt{2}\sigma})) + \frac{\lambda}{2} (1 - \text{erf}(\frac{\lambda}{\sqrt{2}})) \Big). \tag{9}$$

Then the bias caused by the truncation is:

$$\mathbb{E}(\bar{\mathbf{z}}) - \mathbb{E}(\mathbf{z}) = \sigma \Big(-\exp(-\frac{\lambda^2}{2}) + \frac{\lambda}{2} (1 - \text{erf}(\frac{\lambda}{\sqrt{2}})) \Big) = \big(\lambda - \lambda \Phi(\lambda) - \phi(\lambda) \big) \sigma, \tag{10}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ are the probability density function (pdf) and cumulative distribution function (cdf) of the standard normal distribution. One can find that the bias term $\mathbb{E}(\bar{\mathbf{z}}) - \mathbb{E}(\mathbf{z})$ converges to zero as $\lambda \to \infty$. In other words, if λ is large enough, the bias can be very small. Thus, there exists a bias-variance trade-off. See Appendix D for the proof.

A proper selection of λ can improve the detection performance by significantly reducing the uncertainty (variance reduction) and slightly changing the distribution of the features (small bias). If λ is large, $T(\mathbf{x}; \bar{f})$ uses more extreme features in both the ID and OOD data. As λ tends to infinity, $T(\mathbf{x}; \bar{f})$ converges to $T(\mathbf{x}; f)$. Then **BATS** is the same to the original energy detection. If λ is small, extreme features are removed from the test statistic $T(\mathbf{x}; \bar{f})$ while introducing a non-negligible bias. Because of the change in feature distribution, the detection method loses its power to identify OOD examples. Fig. 2 illustrates the distribution of OOD scores with different λ, which empirically verifies this trade-off.

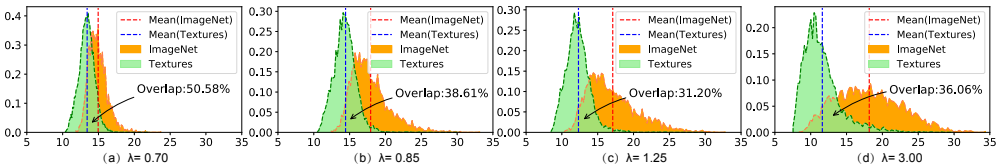


Figure 2: Bias-variance trade-off in BATS. We illustrate the OOD score for ID (ImageNet) and OOD (Textures) examples. Smaller λ contributes to variance reduction which benefits the estimation of the reject region. But smaller λ causes a larger bias, which can drastically alter the distribution of features and damage the performance of the model in distinguishing ID and OOD examples.

5 Experiments

In this section, we first introduce our experiment implementation. Then, we evaluate our methods both on the large-scale OOD detection benchmark [33] and the CIFAR benchmarks [6]. After that, the ablation studies compare the influence of applying rectification on different layers and show the influence of the hyperparameter. Moreover, our **BATS** can also slightly improve the test accuracy of the pre-trained models (in Appendix H). We consider the out-of-distribution detection as a **single-sample** hypothesis testing task and only test one sample at a time.

5.1 Implementation

Dataset. For evaluating the large-scale OOD detection performance, we use ImageNet-1k [33] as the in-distribution dataset and consider four out-of-distribution datasets, including (subsets of) the fine-grained dataset iNaturalist [34], the scene recognition datasets Places [35] and SUN [36], and the texture dataset Textures [37] with non-overlapping categories to ImageNet-1k.

As for the evaluation on CIFAR Benchmarks, we use the CIFAR-10 and CIFAR-100 [38] as the in-distribution datasets using the standard split with 50,000 training images and 10,000 test images. We consider four OOD datasets: SVHN [39], Tiny ImageNet [40], LSUN [41] and Textures [37].

Baselines. We consider different kinds of competitive OOD detection methods as baselines, including Maximum Softmax Probability (MSP) [6], ODIN [7], Energy [5], Mahalanobis [11], GradNorm [12] and ReAct [8]. MSP is a simple baseline for OOD detection and ReAct is a state-of-the-art method that achieves strong detection performance. All methods use the pre-trained networks post-hoc.

Metrics. FPR95: the false positive rate of OOD (negative) examples when the true positive rate of in-distribution (positive) examples is as high as 95%. Lower FPR95 indicates better OOD detection performance and vice versa. **AUROC:** the area under the receiver operating characteristic curve (ROC). Higher AUROC indicates better detection performance. See Appendix E for more details.

5.2 Evaluation on the large-scale OOD detection benchmark

We first evaluate our method on a large-scale OOD detection benchmark proposed by Huang and Li [33]. [33] revealed that OOD detection methods designed for the CIFAR benchmark might not effectively be adaptable for the ImageNet benchmark with a large semantic space. Recent literature [8, 12, 33] proposes to evaluate OOD detection performance on images that have higher resolution and contain more classes than the CIFAR benchmarks, which is more relevant to real-world applications.

In Tab. 1, we compare our method with the existing methods and show the OOD detection performance for each OOD test dataset and the average over the four datasets. We consider different architectures, including the widely used ResNet-50 [42], DenseNet-121 [43] and a lightweight model MobileNet-v2 [44]. Compared with the Energy Score [5], the difference in our approach is rectifying the features that deviate from the feature's typical set. Our method outperforms the Energy Score on ResNet-50 by 31.05% in FPR95 and 8.46% in AUROC. Furthermore, our method reduces FPR95 by 5.11% and improves AUROC by 1.33% compared to the state-of-the-art method [8] on ResNet-50. Here the models are pre-trained in a standard manner. We also show that **BATS** can boost the OOD detection when using the adversarially pre-trained classifiers in Appendix F.

Simultaneously, we observe that existing methods have different performances on different architectures. Specifically, the performance of the GradNorm [12] in FPR95 is 7.86% worse than that of

Table 1: OOD detection performance comparison on different architectures: ResNet-50 (RN50) [42], DenseNet-121 (DN121) [43] and MobileNet-V2 (MNet) [44]. We use the pre-trained models in PyTorch [45] trained on ImageNet. All methods are post hoc and can be directly used for pre-trained models. The best results are in Bold. The up arrow indicates that the higher the value, the better the performance, and vice versa.

Model	Method	iNaturalist		SUN		Places		Textures		Average	
		FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
RN50	MSP[6]	51.44	88.17	72.04	79.95	74.34	78.84	54.90	78.69	63.18	81.41
	ODIN[7]	41.07	91.32	64.63	84.71	68.36	81.95	50.55	85.77	56.15	85.94
	Energy[5]	46.65	91.32	61.96	84.88	67.97	82.21	56.06	84.88	58.16	85.82
	Mahalanobis[11]	97.00	52.65	98.50	42.41	98.40	41.79	55.80	85.01	87.43	55.47
	GradNorm[12]	23.73	93.97	42.81	87.26	55.62	81.85	**38.15**	87.73	40.08	87.70
	ReAct[8]	17.77	96.70	25.15	94.34	34.64	**91.92**	51.31	88.83	32.22	92.95
	BATS(Ours)	**12.57**	**97.67**	**22.62**	**95.33**	**34.34**	91.83	38.90	**92.27**	**27.11**	**94.28**
DN121	MSP[6]	47.65	89.09	69.95	79.64	72.53	78.74	69.69	77.06	64.96	81.13
	ODIN[7]	30.72	93.66	57.90	86.11	63.16	83.54	53.51	83.88	51.32	86.80
	Energy[5]	33.16	93.81	53.79	86.70	61.01	83.83	55.42	84.06	50.85	87.10
	Mahalanobis[11]	97.36	42.24	96.21	41.28	97.32	47.27	62.78	56.53	88.42	46.83
	GradNorm[12]	22.88	94.40	43.12	87.55	55.80	82.00	47.58	85.16	42.35	87.28
	ReAct[8]	15.93	96.91	40.41	90.13	48.87	87.98	36.58	92.48	35.45	91.88
	BATS(Ours)	**14.63**	**97.13**	**30.45**	**93.03**	**41.35**	**89.24**	**31.72**	**93.40**	**29.54**	**93.20**
MNet	MSP[6]	63.09	85.71	79.67	76.01	81.47	75.51	75.12	76.49	74.84	78.43
	ODIN[7]	45.61	91.33	63.03	83.44	70.01	80.85	52.45	85.61	57.78	85.31
	Energy[5]	49.52	91.10	63.06	84.42	69.24	81.42	58.16	84.88	60.00	85.46
	Mahalanobis[11]	62.04	82.37	54.79	86.33	53.77	83.69	88.72	37.28	64.83	72.42
	GradNorm[12]	33.70	92.46	42.15	89.65	56.56	83.93	**34.95**	**90.99**	41.84	89.26
	ReAct[8]	37.08	93.41	53.13	86.04	54.15	83.31	42.45	89.42	46.70	88.05
	BATS(Ours)	**31.56**	**94.33**	**41.68**	**90.21**	**52.43**	**86.26**	38.69	90.76	**41.09**	**90.39**

ReAct [8] on the ResNet-50, but surpasses ReAct on MobileNet-V2 by 4.86%. Our method achieves the best performance on different architectures. Appendix K shows that our method also outperforms the existing methods when choosing the natural adversarial examples [46] as OOD examples.

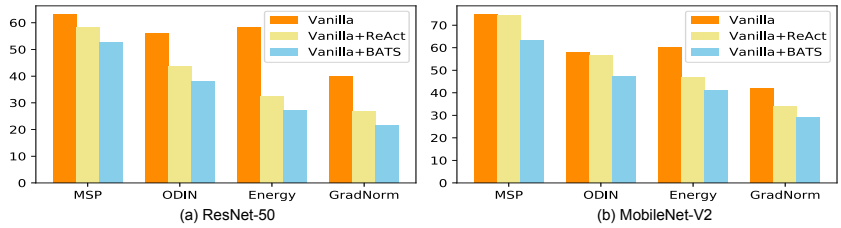


Figure 3: The FPR95 for different methods on ImageNet (lower is better) on ResNet-50 and MobileNet-V2. We illustrate the average performance on four OOD datasets. "Vanilla" means the original method and "Vanilla+BATS" means applying our **BATS** on the method.

Our experiments mainly use the energy score as the test statistic. In Fig. 3, we show that **BATS** is also compatible with various OOD scores and **BATS** can boost the performance of various OOD scores. Applying our **BATS** on GradNorm [12] (a gradient-based OOD score) can even achieve better performance than "Energy+BATS" but this method needs to derive the gradients of the model, which costs more than "Energy+BATS." See Appendix I for detailed performance.

5.3 Evaluation on CIFAR benchmarks

We further evaluate our method on CIFAR benchmarks and use CIFAR-10 and CIFAR-100 [38] as the in-distribution datasets respectively. Tab. 2 compares our method with the baseline methods and shows the OOD detection performance for each OOD test dataset and the average over the four datasets. We evaluate our method on the ResNet-18 (RN18) [42] and WideResNet-28-10 (WRN) [47]. The models are trained for 200 epochs with a batch size of 128. The starting learning rate is 0.1 and decays by a factor of 10 at epochs 100 and 150.

ODIN [7] performs the best in the baselines methods on CIFAR-10 with an FPR95 of 32.99% on ResNet-18. Our method outperforms ODIN by 3.03% and outperforms the simple baseline method MSP [6] by 27.14% in FPR95. As for using CIFAR-100 as the in-distribution dataset, ReAct [8]

Table 2: OOD detection performance on CIFAR-10 and CIFAR-100 [38]. All methods are post hoc and can be directly used for pre-trained models. The best results are in Bold.

Dataset	Method	SVHN		Tiny-Imagenet		LSUN_resize		Texture		Average	
		FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
CIFAR10 RN18	MSP[6]	59.60	91.29	50.01	93.02	52.15	92.73	66.63	88.50	57.10	91.39
	ODIN[7]	59.71	88.52	**10.95**	**98.08**	**9.24**	**98.25**	52.06	89.16	32.99	93.50
	Energy[5]	54.03	91.32	15.18	97.28	23.53	96.14	55.30	89.37	37.01	93.53
	GradNorm[12]	82.45	79.85	19.23	96.77	48.99	90.67	69.40	81.72	55.02	87.25
	ReAct[8]	46.87	92.54	22.80	96.10	18.31	96.92	47.39	91.58	33.84	94.29
	BATS(Ours)	**38.42**	**93.53**	17.75	96.91	19.85	96.59	**43.81**	**92.32**	**29.96**	**94.84**
CIFAR10 WRN	MSP[6]	63.24	86.66	39.57	94.60	44.31	93.82	60.71	88.90	51.96	91.00
	ODIN[7]	61.13	82.49	**12.79**	**97.61**	12.49	97.50	61.13	80.18	36.89	89.45
	Energy[5]	56.05	86.63	17.58	96.99	28.44	95.29	61.74	85.68	40.95	91.15
	GradNorm[12]	88.55	49.14	41.25	90.68	91.02	48.94	90.83	46.28	77.91	58.76
	ReAct[8]	58.35	86.67	18.85	96.62	16.52	97.04	50.89	89.27	36.15	92.40
	BATS(Ours)	**50.60**	**89.50**	25.17	95.66	**11.98**	**97.70**	**45.30**	**91.18**	**33.26**	**93.51**
CIFAR100 RN18	MSP[6]	81.79	77.80	68.32	83.92	82.51	75.73	85.12	73.36	79.44	77.70
	ODIN[7]	**40.82**	**93.32**	69.34	86.28	79.62	82.12	83.61	72.36	68.35	83.52
	Energy[5]	81.24	84.59	40.12	93.16	73.56	82.98	85.87	74.94	70.20	83.92
	GradNorm[12]	57.65	87.77	**25.77**	**95.12**	89.60	63.25	79.08	68.89	63.03	78.76
	ReAct[8]	70.28	88.25	45.62	91.02	55.57	89.32	61.01	87.57	58.12	89.04
	BATS(Ours)	61.48	90.63	44.41	91.27	**52.68**	**90.04**	**52.36**	**89.72**	**52.73**	**90.42**
CIFAR100 WRN	MSP[6]	78.43	77.74	61.33	87.46	81.69	72.69	85.07	75.46	76.63	78.34
	ODIN[7]	**35.69**	**94.84**	82.68	79.17	87.48	74.53	86.97	65.40	73.21	78.49
	Energy[5]	75.57	83.05	**40.87**	**92.99**	65.90	82.78	87.98	71.21	67.58	82.51
	GradNorm[12]	83.24	72.55	45.20	90.43	78.62	68.80	92.59	46.99	74.91	69.69
	ReAct[8]	72.94	86.89	42.07	91.97	60.87	85.90	84.18	76.22	65.02	85.25
	BATS(Ours)	71.01	87.50	41.93	91.97	**57.01**	**88.04**	**80.46**	**78.42**	**62.60**	**86.48**

is the best baseline method. Our approach surpasses the ReAct by 5.39% in FPR95 on ResNet-18. Our method achieves the best performance on both CIFAR-10 and CIFAR-100. Our approach is also effective when using the WideResNet model, outperforming the existing methods.

5.4 Ablation studies

5.4.1 Rectifying the features of the early layers

In our experiments, we rectify the features of the penultimate layer (the layer before the fully connected layer), which is convenient and efficient. However, **what will happen if we rectify the features of the early layers with BATS?** The early layers refer to the layers close to the input [48]. In particular, the original ResNet-50 [42] consists of four residual blocks. Block1 is close to the input and Block4 is close to the output. In Tab. 3, we show the influence of applying feature rectification on the output of different blocks. Applying feature rectification to the early blocks (from Block1 to Block3) has little effect on the performance of OOD detection, while the last block plays a vital role. Applying feature rectification on all the blocks performs the best in our experiments, which is 0.99% higher than the "Block4" in FPR95 and 32.04% higher than "Without" in FPR95. Considering that the latest block has a more significant impact on the OOD detection performance than the other blocks, we just rectify the features of the penultimate layer with **BATS** for the simplicity of the method.

Table 3: Ablation study of the influence of feature rectification on different blocks. "Without" means applying no rectification on any blocks. "Block1-4" means applying rectification on all blocks.

Blocks	iNaturalist		SUN		Places		Textures		Average	
	FPR95	AUROC	FPR95	AUROC	FPR95	AUROC	FPR95	AUROC	FPR95	AUROC
Without	46.65	91.32	61.96	84.88	67.97	82.21	56.06	84.88	58.16	85.82
Block1	49.72	90.73	62.67	84.62	68.30	82.03	55.62	85.04	59.08	85.61
Block2	41.78	92.36	63.73	84.67	69.45	81.95	55.53	85.45	57.62	86.11
Block3	40.76	92.55	58.37	86.56	64.78	83.82	51.45	86.77	53.84	87.43
Block4	**12.57**	**97.67**	22.62	95.33	34.34	91.83	38.90	92.27	27.11	94.28
Block1-2	43.63	91.92	63.22	84.61	69.28	81.84	53.67	85.72	57.45	86.02
Block1-3	38.05	93.04	59.47	86.47	66.30	83.49	49.72	87.50	53.39	87.63
Block1-4	12.76	97.54	**21.15**	**95.51**	**33.01**	**91.91**	**37.55**	**92.54**	**26.12**	**94.38**

To find out why the last block has a significant influence on the OOD detection while the other blocks contribute little, we visualize the feature embeddings extracted by different blocks in ResNet-50 using t-SNE [49] in Fig. 4. We choose the iNaturalist as the OOD dataset and the ImageNet as the ID dataset. The features extracted by the early blocks of the ID and OOD examples are similar, which

has little benefit in distinguishing the ID and OOD examples. In contrast, the last block can extract perfectly separable features for the ID and OOD examples. This may be due to the fact that deep neural networks focus on similar general features (edges, lines, and colors) in the early layers and pay more attention to specific features related to classification in the late layers [48, 50]. The late layer can contribute more to the OOD detection than the early layer. See more in Appendix B.

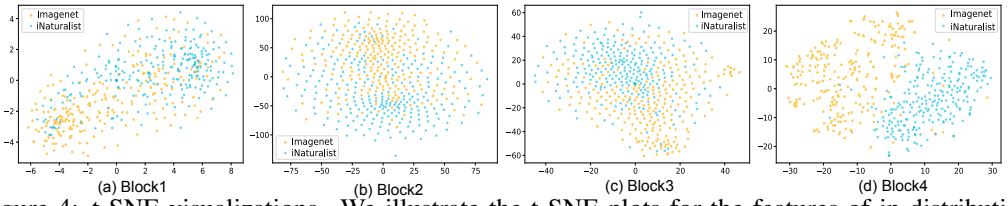


Figure 4: t-SNE visualizations. We illustrate the t-SNE plots for the features of in-distribution examples (ImageNet) and out-of-distribution examples (iNaturalist) from different blocks.

5.4.2 The influence of the hyperparameter

In Sec. 4.3, we theoretically analyze the bias-variance trade-off in our method. Our proposed **BATS** can reduce variance, which benefits OOD detection, but can also introduce a bias. Here, we empirically show the influence of the hyperparameter λ in Fig. 5. As λ tends to infinity, BATS approaches to the Energy Score (the horizontal lines). Very small λ will damage the performance.

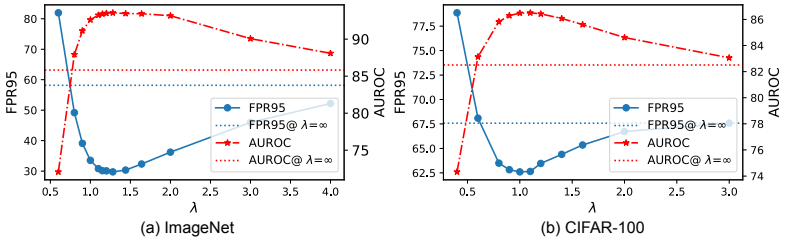


Figure 5: (a) The influence of the hyper-parameter λ on the OOD detection on ImageNet. The model is ResNet-50. We illustrate the average performance on four OOD datasets. (b) The influence of the hyper-parameter λ on the OOD detection on CIFAR-100. The model is WideResNet. The horizontal line indicates the OOD detection performance without feature rectification.

6 Conclusion

In this paper, we provide novel insight into the obstacle factor in OOD detection from the perspective of typicality and hypothesize that extreme features can be the culprit. We propose to rectify the features into the typical set and provide a concise and effective post-hoc approach **BATS** to estimating the feature's typical set. **BATS** can be applied to various OOD scores to boost the OOD detection performance. Theoretical analysis and ablations provide a further understanding of our approach. Experimental results show that our **BATS** can establish state-of-the-art OOD detection performance on the ImageNet benchmark, surpassing the previous best method by 5.11% in FPR and 1.43% in AUROC. We hope that our findings can motivate new research into the internal mechanisms of deep models and OOD detection and uncertainty estimation from the perspective of feature typicality.

Limitations and societal impact. This paper proposes to rectify the feature into its typical set to improve the detection performance against OOD data and provides a plug-and-play method with the assistance of BN. The limitation of our method can be that the BN layers are required in the model architecture in our approach. BN layers are widely used in convolutional neural networks to alleviate covariate shifts, but there are also architectures without BN. A set of training images can contribute to selecting the feature's typical set and alleviate this limitation. We also anticipate some other information in the model is conducive to selecting the feature's typical set and improving the post-hoc OOD detection performance. We leave this as future work. Regarding potential negative impact, truncating the features into the typical set can improve the OOD detection but will introduce a bias and inherently cause some information loss which may be important for the model in the real-world scenario.

References

[1] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[2] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37:100270, 2020.

[3] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[4] Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer. Deep learning for financial applications: A survey. *Applied Soft Computing*, 93:106384, 2020.

[5] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, 2020.

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

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

[8] Yiyou Sun, Chuan Guo, and Yixuan Li. React: Out-of-distribution detection with rectified activations. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[9] Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[10] Aristotelis-Angelos Papadopoulos, Mohammad Reza Rajati, Nazim Shaikh, and Jiamian Wang. Outlier exposure with confidence control for out-of-distribution detection. *Neurocomputing*, 441:138–150, 2021.

[11] 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*, volume 31. Curran Associates, Inc., 2018.

[12] Rui Huang, Andrew Geng, and Yixuan Li. On the importance of gradients for detecting distributional shifts in the wild. In *Advances in Neural Information Processing Systems*, 2021.

[13] Haoran Wang, Weitang Liu, Alex Bocchieri, and Yixuan Li. Can multi-label classification networks know what they don't know? In *Advances in Neural Information Processing Systems*, 2021.

[14] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019.

[15] Julian Bitterwolf, Alexander Meinke, and Matthias Hein. Certifiably adversarially robust detection of out-of-distribution data. *Advances in Neural Information Processing Systems*, 33:16085–16095, 2020.

[16] Dan Hendrycks*, Norman Mu*, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple method to improve robustness and uncertainty under data shift. In *International Conference on Learning Representations*, 2020.

[17] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[18] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2018.

[19] Jiefeng Chen, Yixuan Li, Xi Wu, Yingyu Liang, and Somesh Jha. Atom: Robustifying out-of-distribution detection using outlier mining. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 430–445. Springer, 2021.

[20] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

[21] Ev Zisselman and Aviv Tamar. Deep residual flow for out of distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13994–14003, 2020.

[22] Joan Serrà, David Álvarez, Vicenç Gómez, Olga Slizovskaia, José F Núñez, and Jordi Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. In *International Conference on Learning Representations*, 2019.

[23] Zhisheng Xiao, Qing Yan, and Yali Amit. Likelihood regret: An out-of-distribution detection score for variational auto-encoder. *Advances in neural information processing systems*, 33:20685–20696, 2020.

[24] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2018.

[25] Polina Kirichenko, Pavel Izmailov, and Andrew G Wilson. Why normalizing flows fail to detect out-of-distribution data. *Advances in neural information processing systems*, 33:20578–20589, 2020.

[26] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, and Balaji Lakshminarayanan. Detecting out-of-distribution inputs to deep generative models using typicality. *arXiv preprint arXiv:1906.02994*, 2019.

[27] Amirhossein Ahmadian and Fredrik Lindsten. Likelihood-free out-of-distribution detection with invertible generative models. *In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2021.

[28] Matan Haroush, Tzviel Frostig, Ruth Heller, and Daniel Soudry. A statistical framework for efficient out of distribution detection in deep neural networks. In *International Conference on Learning Representations*, 2022.

[29] Lily Zhang, Mark Goldstein, and Rajesh Ranganath. Understanding failures in out-of-distribution detection with deep generative models. In *International Conference on Machine Learning*, pages 12427–12436. PMLR, 2021.

[30] Federico Bergamin, Pierre-Alexandre Mattei, Jakob Drachmann Havtorn, Hugo Senetaire, Hugo Schmutz, Lars Maaløe, Soren Hauberg, and Jes Frellsen. Model-agnostic out-of-distribution detection using combined statistical tests. In *International Conference on Artificial Intelligence and Statistics*, pages 10753–10776. PMLR, 2022.

[31] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *International Conference on Learning Representations*, 2020.

[32] Yao Zhu, Jiacheng Ma, Jiacheng Sun, Zewei Chen, Rongxin Jiang, Yaowu Chen, and Zhenguo Li. Towards understanding the generative capability of adversarially robust classifiers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7728–7737, 2021.

[33] Rui Huang and Yixuan Li. Mos: Towards scaling out-of-distribution detection for large semantic space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8710–8719, 2021.

[34] 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.

[35] 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.

[36] Jianxiong Xiao, J Hays, KA Ehinger, A Oliva, and A Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

[37] 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.

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

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

[40] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[41] 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.

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[43] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[44] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32*, pages 8024–8035, 2019.

[46] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15262–15271, 2021.

[47] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

[48] Johannes Mehrer, Courtney Spoerer, Nikolaus Kriegeskorte, and Tim Kietzmann. Individual differences among deep neural network models. *Nature Communications*, 01 2020. doi: 10.1101/2020.01.08.898288.

[49] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[50] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[51] Jerzy Neyman and Egon Sharpe Pearson. Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933.

[52] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning, 2006.

[53] Andrew W. Palmer, Andrew J. Hill, and Steven J. Scheding. Methods for stochastic collection and replenishment (scar) optimisation for persistent autonomy. *Robotics and Autonomous Systems*, 87:51–65, 2017. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2016.09.011.

[54] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems*, 33:3533–3545, 2020.

[55] Alireza Shafaei, Mark Schmidt, and James J Little. A less biased evaluation of out-of-distribution sample detectors. In *BMVC*, 2019.

[56] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018.

[57] Daniel Omeiza, Skyler Speakman, Celia Cintas, and Komminist Weldermariam. Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models. *arXiv preprint arXiv:1908.01224*, 2019.

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

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

[60] 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.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See abstract and Section 1.
   (b) Did you describe the limitations of your work? [Yes] We describe the limitation of our work in our Conclusion section.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We discuss the potential negative societal impacts of our work in the Conclusion section.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 4.3. We theoretically analyze the bias-variance trade-off and state the full set of assumptions.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix D
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We introduce the implementation of our method in Section 5.1 and Appendix E. We plan to open the source code to reproduce the main experimental results later.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the Section 5.1 and Appendix E for details.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix E.3
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use the pre-trained model in PyTorch and cite the creators.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Type I error and type II error in OOD detection

In the preliminary section, we provide a summary for the out-of-distribution detection from the perspective of hypothesis testing. As for the error of an OOD detection method, it can be evaluated from two dimensions. The mistaken rejection of an actually true null hypothesis $\mathcal{H}_0$ is the type I error. The significance level α is a predetermined scalar that bounds the type I error above:

$$\alpha >= P(\mathbf{x} \in \mathcal{R}|\mathcal{H}_0) = P(T(\mathbf{x}; f) \geq \gamma|\mathcal{H}_0) = P_0(T(\mathbf{x}; f) \geq \gamma).$$

By the Neyman–Pearson lemma [51], the threshold γ is determined by solving the equation $\alpha = P_0(T(\mathbf{x}; f) \geq \gamma)$. For a given significance level, the goal is to minimize the type II error: the failure to reject a null hypothesis that is actually false. The probability of the type II error is denoted by

$$\beta = P(x \notin \mathcal{R}|\mathcal{H}_1) = P(T(\mathbf{x}; f) < \gamma|\mathcal{H}_1).$$

In the literature on OOD detection, the type II error is also denoted by "FPR$(1-\alpha)$", which is short for "the false positive rate of OOD examples when the true positive rate for ID examples is $(1-\alpha)\%$." In the experiments, we follow the notation FPR$(1-\alpha)$.

In our paper, we mainly show the superiority of our method on different datasets in the metrics of FPR95 and AUROC. Here we illustrate the OOD detection performance at different significance levels (FPR(1-α)) in Fig. 6. The horizontal axis represents the significance level for FPR ("0.95" means FPR95). Our method surpasses the existing methods at different significance levels on both the large scale dataset (ImageNet) and the small scale dataset (CIFAR-10).

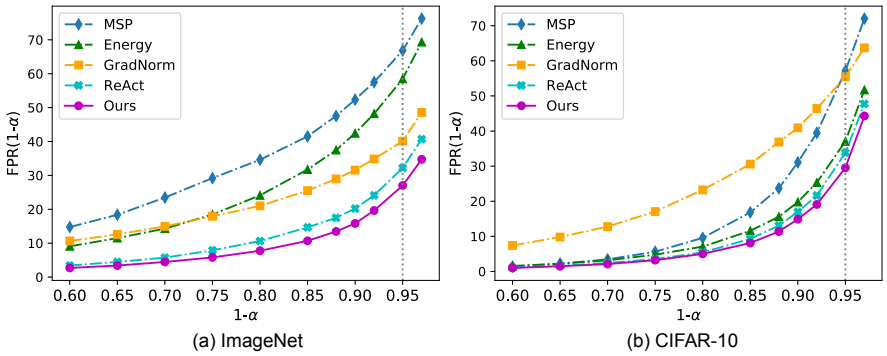


Figure 6: (a) The FPR(1-α) for different methods on ImageNet (lower is better). The model is ResNet-50. We illustrate the average performance on four OOD datasets. The grey vertical line indicates the performance in FPR95. (b) The FPR(1-α) for different methods on CIFAR-10. The model is ResNet-18.

B The influence of the early layers

In the paper, we show that the early layers of the model can hardly distinguish the feature embeddings of the in-distribution examples (ImageNet-1k) and out-of-distribution examples (iNaturalist) for that the features of these examples extracted by the early layers are mixed up. Rectifying the features of the early layers contributes little to OOD detection. In this section, we illustrate the t-SNE visualization for the feature embeddings of in-distribution examples and other out-of-distribution examples (Places [35], SUN [36], and Textures [37]) from different blocks. Their t-SNE visualization results are similar. To be specific, the feature embeddings of the early blocks of the different datasets are similar, while the last block shows differences. In Tab. 3, we set the λ for Block1 and Block2 as 5, the λ for Block3 as 2, and the λ for Block4 as 1, for that restricting the features of the early layers may have a negative impact on the late layers.

C Energy and density in the classifier

To make our paper self-contained, we provide some details for the energy and density in the classifier with reference to the previous works [32, 31, 5].

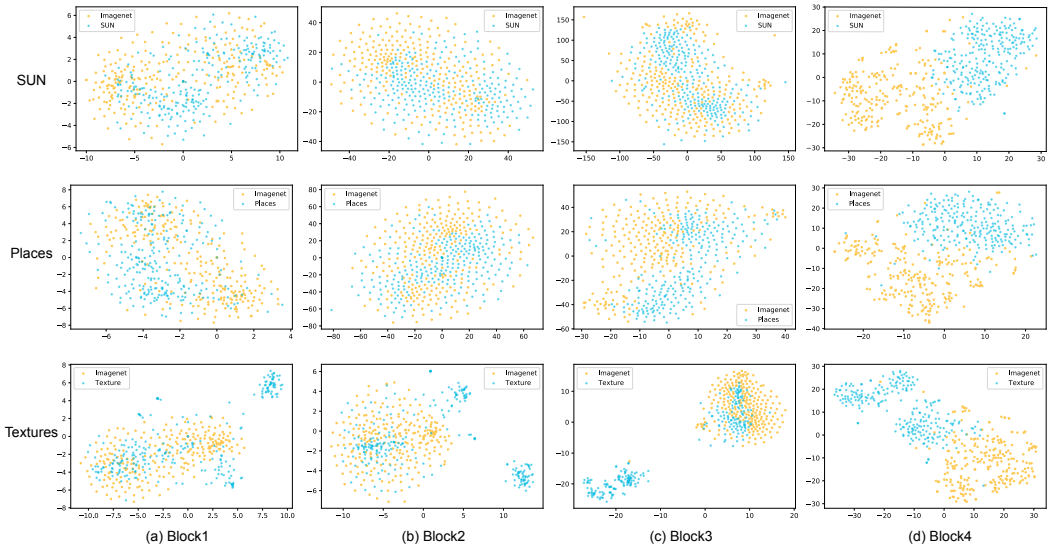


Figure 7: t-SNE visualization for the feature embeddings of in-distribution examples and out-of-distribution examples from different blocks. The model we used is ResNet-50.

LeCun et al. [52] show that any probability density $p(x)$ for x can be expressed as

$$p(x) = \frac{\exp(-E(x))}{Z}, \tag{11}$$

where $E(x)$ represents the energy of x and is modeled by neural network, $Z = \int \exp(-E(x))dx$ is the normalizing factor which is also known as the partition function.

Similarly, $p(x, y)$ can be defined as follows:

$$p(x, y) = \frac{\exp(-E(x, y))}{\tilde{Z}}, \tag{12}$$

where $\tilde{Z} = \int \sum_y \exp(-E(x, y))dx$.

Thus we also get $p(y|x)$ expressed by $E(x)$ and $E(x, y)$:

$$p(y|x) = \frac{p(x, y)}{p(x)} = \frac{\exp(-E(x, y)) \cdot Z}{\exp(-E(x)) \cdot \tilde{Z}}. \tag{13}$$

We denote f as a classification neural network. Let x be a sample. Then $f(x)[k]$ represents the kth output of the last layer and $p(y|x)$ can be defined as:

$$p(y|x) = \frac{\exp(f(x)[y])}{\sum_{k=1}^{n} \exp(f(x)[k])}, \tag{14}$$

where n represents total possible classes. From Eq. (13) and (14), we define two energy functions as follows:

$$\begin{cases} E(x, y) = -\log(\exp(f(x)[y])), \\ E(x) = -\log(\sum_{k=1}^{n} \exp(f(x)[k])). \end{cases} \tag{15}$$

And thus Z can be expressed as:

$$Z = \int_x \exp(-E(x))dx = \int_x \exp(\log(\sum_y \exp(f(x)[y])))dx = \int_x (\sum_y \exp(f(x)[y]))dx = \tilde{Z}. \tag{16}$$

From Eq. (15) and Eq. (11), the marginal density $p(x)$ for x can be expressed by the output of the classifier as:

$$p(x) = \frac{\sum_{k=1}^{n} \exp(f(x)[k])}{Z}, \tag{17}$$

where Z is independent to $\mathbf{x}$.

D Proofs of section 4.3

In this section, we prove the main results in Section 4.3. Recall the layer structure in (3):

$$\mathbf{z}' \to \text{BN}(\mathbf{z}'; \mu, \sigma) \text{ or } \text{TrBN}(\mathbf{z}'; \mu, \sigma, \lambda) \to \text{ReLU} \to \mathbf{z}, \tag{18}$$

where $\mathbf{z}'$ is the feature vector extracted from the penultimate layer of g. We denote

$$\mathbf{z}_1 = \text{BN}(\mathbf{z}'; \mu, \sigma) \quad \text{and} \quad \bar{\mathbf{z}}_1 = \text{TrBN}(\mathbf{z}'; \mu, \sigma). \tag{19}$$

Suppose $\mathbf{z}'$ is a Gaussian variable. Then $\mathbf{z}_1 \sim N(\mu, \sigma^2)$ and $\bar{\mathbf{z}}_1$ follows a Rectified Gaussian distribution with lower bound $\mu - \lambda\sigma$ and upper bound $\mu + \lambda\sigma$. The cdf of $\bar{\mathbf{z}}_1$ is

$$F^R(\mathbf{z}'|\mu, \sigma^2) = \begin{cases} 0, & \text{if} \quad \mathbf{z}' < a; \\ \Phi(\mathbf{z}'; \mu, \sigma^2), & \text{if} \quad a \le \mathbf{z}' < b; \\ 1, & \text{if} \quad b \le \mathbf{z}', \end{cases} \tag{20}$$

where $\Phi(\mathbf{z}'; \mu, \sigma^2)$ the cdf of a normal distribution with mean μ and variance σ^2. According to [53],

$$\mathbb{E}(\bar{\mathbf{z}}_1) = \mu \quad \text{and} \quad \text{Var}(\bar{\mathbf{z}}_1) = \sigma^2 C(\lambda), \tag{21}$$

where

$$C(\lambda) = \text{erf}(\frac{\lambda}{\sqrt{2}}) - \frac{\sqrt{2}}{\sqrt{\pi}}\lambda \exp(-\frac{\lambda^2}{2}) + \lambda^2(1 - \text{erf}(\frac{\lambda}{\sqrt{2}})), \tag{22}$$

and $\text{erf}(x) = (2/\sqrt{\pi}) \int_0^x \exp(-t^2)dt$ is the Gauss error function. It is easy to see

$$C(0) = 0 \quad \text{and} \quad C'(\lambda) = 2\lambda \cdot \text{erf}(\frac{\lambda}{\sqrt{2}}) > 0. \tag{23}$$

In addition,

$$\begin{aligned} \lambda^2(1 - \text{erf}(\frac{\lambda}{\sqrt{2}})) &= \lambda^2 \frac{2}{\sqrt{\pi}} \int_{\lambda/\sqrt{2}}^{+\infty} \exp(-t^2)\mathrm{d}t \\ &\le \frac{4}{\sqrt{\pi}} \int_{\lambda/\sqrt{2}}^{+\infty} t^2 \exp(-t^2)\mathrm{d}t \to 0, \quad \lambda \to +\infty. \end{aligned} \tag{24}$$

Therefore, as λ tends to $+\infty$,

$$\text{erf}(\frac{\lambda}{\sqrt{2}}) \to 1, \quad \frac{\sqrt{2}}{\sqrt{\pi}}\lambda \exp(-\frac{\lambda^2}{2}) \to 0, \quad \lambda^2(1 - \text{erf}(\frac{\lambda}{\sqrt{2}})) \to 0. \tag{25}$$

We obtain that $C(\lambda) \to 1$ as $\lambda \to +\infty$.

Next we deal with the bias term. To proceed further, we need more notations as follow:

$$\mathbf{z} = \text{ReLU}(\text{BN}(\mathbf{z}'; \mu, \sigma)) \quad \text{and} \quad \bar{\mathbf{z}} = \text{ReLU}(\text{TrBN}(\mathbf{z}'; \mu, \sigma)). \tag{26}$$

Then we know that $\mathbf{z}$ follows a Rectified Gaussian distribution with lower bound 0 and upper bound $+\infty$ and $\bar{\mathbf{z}}$ is a Rectified Gaussian variable with lower bound 0 and upper bound $\mu + \lambda\sigma$. According to [53], their expectations are

$$\mathbb{E}(\mathbf{z}) = \mu + \sigma\Big(\frac{1}{\sqrt{2\pi}}\big(\exp(-\frac{\mu^2}{2\sigma^2})\big) - \frac{\mu}{2\sigma}(1 + \text{erf}(-\frac{\mu}{\sqrt{2}\sigma}))\Big), \tag{27}$$

and

$$\begin{aligned} \mathbb{E}(\bar{\mathbf{z}}) &= \mu + \sigma\Big(\frac{1}{\sqrt{2\pi}}\big(\exp(-\frac{\mu^2}{2\sigma^2}) - \exp(-\frac{\lambda^2}{2})\big) \\ &\quad - \frac{\mu}{2\sigma}(1 + \text{erf}(-\frac{\mu}{\sqrt{2}\sigma})) + \frac{\lambda}{2}(1 - \text{erf}(\frac{\lambda}{\sqrt{2}}))\Big). \end{aligned} \tag{28}$$

Therefore the bias term is

$$\mathbb{E}(\bar{\mathbf{z}}) - \mathbb{E}(\mathbf{z}) = \sigma\Big(- \exp(-\frac{\lambda^2}{2}) + \frac{\lambda}{2}(1 - \text{erf}(\frac{\lambda}{\sqrt{2}}))\Big). \tag{29}$$

In addition,

$$\begin{aligned}\frac{\lambda}{2}(1-\text{erf}(\frac{\lambda}{\sqrt{2}})) &= \frac{\lambda}{2}\frac{2}{\sqrt{\pi}}\int_{\lambda/\sqrt{2}}^{+\infty}\exp(-t^2)\mathrm{d}t \\ &\leq \frac{\sqrt{2}}{\sqrt{\pi}}\int_{\lambda/\sqrt{2}}^{+\infty} t\exp(-t^2)\mathrm{d}t \to 0 \quad \text{as} \quad \lambda \to +\infty.\end{aligned} \tag{30}$$

Then we obtain that

$$\text{Bias} = \mathbb{E}(\bar{\mathbf{z}}) - \mathbb{E}(\mathbf{z}) \to 0 \quad \text{as} \quad \lambda \to +\infty. \tag{31}$$

E Experiments details

E.1 Details for metrics

FPR95: the false positive rate of OOD (negative) examples when the true positive rate of in-distribution (positive) examples is as high as 95$\%$. The **t**rue **p**ositive **r**ate (TPR) can be computed as:

$$TPR = \frac{TP}{(TP+FN)}, \tag{32}$$

where TP denotes the true positive (correctly identify the in-distribution examples as in-distribution examples) and FN denotes the False Negative (incorrectly identity the in-distribution examples as out-of-distribution examples). The **f**alse **p**ositive **r**ate (FPR) can be computed as:

$$FPR = \frac{FP}{(FP+TN)}, \tag{33}$$

where FP denotes the false positive (incorrectly identify the out-of-distribution examples as in-distribution examples) and TN denotes the true negative (correctly identify the out-of-distribution examples as out-of-distribution examples).

AUROC: the area under the receiver operating characteristic curve (ROC) which is the plot of TPR vs FPR. If FPR = 0 and TPR = 1, it means that this is a perfect OOD detector, which identify all examples correctly. If FPR=1 and TPR=0, this is a terrible detector that can not make any correct prediction. The closer the area under the ROC curve is to 1, the better the performance of the detector.

E.2 Details for datasets

E.2.1 Large-scale OOD detection

We use the subsets from the following datasets as OOD examples and follow the setting in [8] and [12]. The subsets are curated to be disjoint from the ImageNet-1k labels. We set $\lambda = 1.25$ in our approach for ResNet-50 and DenseNet-121, and $\lambda = 0.4$ for MobileNet-V2. During the evaluation, all images are resized to $3 \times 224 \times 224$.

iNaturalist [34]: iNaturalist contains 675,170 training and validation images from 5089 natural fine-grained categories, including 13 major categories such as plants, insects, birds, and mammals. We randomly sample 10000 images that are disjoint from ImageNet-1k for evaluation.

Places [35]: Places is a scene image dataset, which contains 10 million pictures and more than 400 different types of scene environments. We randomly sample 10000 images that are disjoint from ImageNet-1k for evaluation.

SUN [36]: The Scene UNderstanding (SUN) contains 397 well-sampled categories to evaluate the performance of scene recognition algorithms. We randomly sample 10000 images that are disjoint from ImageNet-1k for evaluation.

Textures [37]: Describable Textures Dataset (DTD) is a texture dataset, including 5640 images, which can be divided into 47 categories according to human perception. We use the entire Textures dataset for evaluation.

E.2.2 CIFAR OOD detection

We use the CIFAR-10 and CIFAR-100 [38] as the in-distribution examples respectively. CIFAR-10 consists of 60,000 images in the shape of $3 \times 32 \times 32$, including 10 categories (aircraft, cars, birds, cats, deer, dogs, frogs, horses, boats, and trucks). CIFAR-100 contains 100 categories of images, and each category has 600 images in the shape of $3 \times 32 \times 32$. We evaluate our approach on four common OOD datasets. We set $\lambda = 3$ in our approach for CIFAR-10 and $\lambda = 1.5$ and for CIFAR-100 on ResNet-18. As for WideResNet-28-10, we set $\lambda = 0.7$ for CIFAR-10 and $\lambda = 1.0$ for CIFAR-100. During the evaluation, all images are resized to $3 \times 32 \times 32$.

SVHN [39]: Street View House Number (SVHN) consists of the house numbers extracted from Google Street View images. We use the entire of its test set as OOD examples (26032 images).

Tiny ImageNet [40]: Similar to ImageNet, Tiny ImageNet is an image classification dataset, which contains 200 categories, and each category contains 50 test images. We randomly crop the images to $3 \times 32 \times 32$.

LSUN [41]: LSUN is a scene understanding dataset, which mainly includes scene images of bedrooms, fixed houses, living rooms, classrooms, etc. We randomly sample 10000 images as out-of-distribution examples and resize the images to $3 \times 32 \times 32$.

Textures [37]: Describable Textures Dataset (DTD) is a texture dataset, including 5640 images, which can be divided into 47 categories according to human perception. We use the entire Textures dataset for evaluation.

E.3 Hardware

Our experiments are implemented by PyTorch [45] and runs on RTX-2080TI.

F Boosting the OOD detection on the robust classifiers

The models used in our experiments (Tab. 1) are standard pre-trained. Salman et al. [54] show that adversarially robust models with less accuracy often perform better than their standard-trained counterparts in transfer learning. In Fig. 8 we evaluate the OOD detection performance on eight adversarially pre-trained ResNet-50 trained with ℓ_2 perturbation of different strength ϵ [54] on the ImageNet benchmark. The horizontal axis represents the perturbation strength in training the model, e.g. "0.05" represents the robust model trained with ℓ_2 perturbation $\epsilon = 0.05$. The strength of perturbation has a great influence on GradNorm [12], but little influence on other methods. Our BATS surpasses all the existing methods on different robust models.

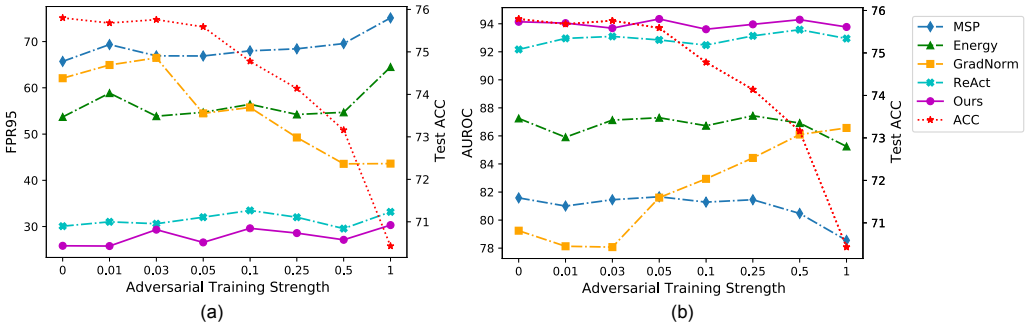


Figure 8: (a) The FPR95 for different methods on ImageNet benchmark (lower is better) on different robust models. We illustrate the average performance on four OOD datasets. The red dotted line indicates the test accuracy of the robust model. (b) The AUROC for different methods (higher is better).

G Related literature to OOD detection

OOD detection has received wide attention because it is critical to ensuring the reliability and safety of deep neural networks. The literature related to OOD detection can be broadly grouped into the following themes. Our paper briefly reviews the literature related to post-hoc detection methods. Here, we provide a more comprehensive review.

Post-hoc Detection Methods. Post-hoc methods focus on improving the OOD uncertainty estimation by utilizing the pre-trained classifiers rather than retraining a model, which is beneficial for adopting OOD detection in real-world scenarios and large-scale settings. In this paper, we mainly focus on the post-hoc OOD detection methods. Hendrycks and Gimpel [6] observe that the maximum softmax probability of In-Distribution Examples (ID) can be higher than the Out-of-Distribution (OOD) samples and provide a simple baseline for OOD detection. ODIN [7] introduces a large sufficiently temperature factor and input perturbation to separate the ID and OOD examples. Liu et al. [5] analyze the limitations of softmax function in OOD detection and propose to use energy score as an indicator. The examples with high energy are considered as OOD examples, and vice versa. Wang et al. [13] propose to use joint energy score which take labels into consideration to enhance the OOD detection. ReAct [8] hypothesize that the OOD examples can trigger the abnormal activation of the model, and propose to clamp the activation value larger than the threshold value to improve the detection performance. Lee et al. [11] use the mixture of Gaussians to model the distribution of feature representations and propose using the feature-level Mahalanobis distance instead of the model output. GradNorm [12] shows that the gradients of the categorical cross-entropy loss contains useful information for OOD detection.

Confidence Enhancement Methods. To enhance the sensitivity to OOD examples, some methods propose to introduce the adversarial examples into the training process. Hein et al. [14] endow low confidence predictions to the examples far away from the training data through an adversarial training optimization technique. Moreover, Bitterwolf et al. [15] enforce low confidence in an l2 ball around the OOD examples. Proper data augmentation also contributes to OOD uncertainty estimation [16, 17, 9]. Some methods take advantage of a set of collected OOD examples to enhance the uncertainty estimation, which are named outlier exposure methods [18, 10, 19]. The correlations between the collected and real OOD examples can largely affect the performance of outliers exposure methods [55].

Density-based Methods. Directly estimating the density of the examples can be a natural approach, this kind of methods explicitly model the distribution of ID examples with probabilistic models and distinguish the OOD/ID examples through the likelihood [20–23]. However, some works show that the probabilistic models may assign higher likelihood to OOD examples than ID examples and fail to distinguish OOD/ID examples [24, 25].

H Test accuracy

In this section, we show that with a proper hyper-parameter λ, our feature rectification method can slightly improve the test accuracy of the pre-trained models both on the clean images and the corrupted images. Here we set $\lambda = 3$. As shown in Tab. 4, we evaluate the test accuracy of the normal pre-trained models and the pre-trained models with our feature rectification method on the clean images and the corrupted images. We choose some image corruption methods used in [56], including salt-and-pepper noise (SP(0.2)), cropout (Crop(0.8)), JPEG compression (JPEG(50)), Gaussian blur (GB) and Gaussian noise (GN). Fig. 9 shows some examples that can be classified correctly by our feature rectified ResNet-50 but classified wrongly by the original ResNet-50.

Table 4: Test accuracy on ImageNet with the pre-trained ResNet-50 and DenseNet-121. Our method rectifies the feature vector of the model and performs well on both the clean images and the corrupted images. The best results are in bold.

Model	Method	Vanilla	SP(0.2)	Crop(0.6)	JPEG(50)	GB(2)	GB(3)	GN(0.5)	GN(1)
RN50	Normal	74.548	40.436	65.488	58.426	52.762	49.022	28.638	10.864
	Ours	**74.610**	**40.506**	**65.770**	**58.454**	**52.920**	**49.312**	**28.852**	**11.014**
DN121	Normal	71.956	43.078	63.150	61.298	50.362	46.980	40.600	25.786
	Ours	**72.050**	**43.146**	**63.498**	**61.332**	**50.520**	**47.014**	**40.618**	**25.852**

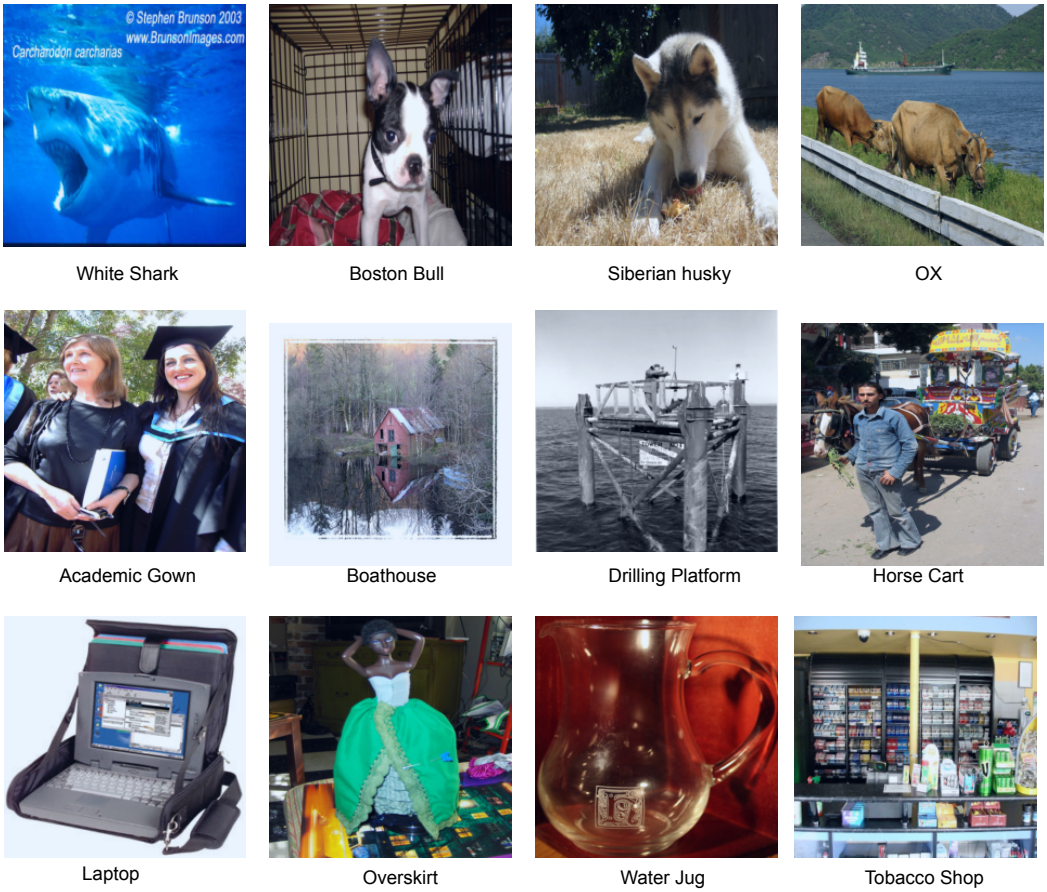


Figure 9: We illustrate some examples in ImageNet that can be classified correctly by the feature rectified ResNet-50 but classified wrongly by the original ResNet-50.

I BATS on other detection methods

In our paper, we provide a concise and effective approach **BATS** to improve the performance of the existing OOD detection methods. We mainly show the effectiveness of applying our feature rectification (BATS) on Energy Score [5]. In Tab. 5 we show that out method is compatible with various OOD detection test statistics (including output-based methods [6, 5, 7] and gradient-based method [12]) and can bring improvements to different methods. Applying our **BATS** on GradNorm can even achieve better performance than "Energy+**BATS**" but this method needs to derive the gradients of the model which cost more than "Energy+**BATS**".

OOD detection methods hope to assign higher scores to the in-distribution (ID) examples and lower scores to the out-of-distribution (OOD) examples. The advanced detection method (GradNorm [12]) can assign better scores to ID and OOD examples than the simple baseline method (MSP [6]). However, there still exists an overlap in the distribution of the scores. As shown in Fig. 10, our BATS reduces the variance of the scores and makes the scores of the ID and OOD examples more separable, which can improve the performance of the OOD detection methods. We think combining our method with a better OOD score can achieve better performance. “BATS+Energy" has already achieved state-of-the-art performance on large-scale and small-scale benchmarks.

Table 5: Applying feature rectification (**BATS**) and ReAct [8] on different OOD detection methods. The best results are in bold.

Model	Method	iNaturalist		SUN		Places		Textures		Average	
		FPR95	AUROC	FPR95	AUROC	FPR95	AUROC	FPR95	AUROC	FPR95	AUROC
RN50	MSP	51.44	88.17	72.04	79.95	74.34	78.84	54.90	78.69	63.18	81.41
	MSP+ReAct	44.90	91.68	60.86	86.22	64.95	84.48	62.06	85.46	58.19	86.96
	MSP+**BATS**	35.79	93.56	56.97	88.08	63.24	85.35	55.14	87.93	52.79	88.73
	ODIN	41.07	91.32	64.63	84.71	68.36	81.95	50.55	85.77	56.15	85.94
	ODIN+ReAct	32.10	93.84	45.14	90.34	52.48	87.92	45.07	87.95	43.70	90.01
	ODIN+**BATS**	25.43	95.44	40.12	92.28	50.57	88.87	36.67	92.42	38.20	92.25
	Energy	46.65	91.32	61.96	84.88	67.97	82.21	56.06	84.88	58.16	85.82
	Energy+ReAct	17.77	96.70	25.15	94.34	34.64	91.92	51.31	88.83	32.22	92.95
	Energy+**BATS**	12.57	97.67	22.62	95.33	34.34	91.83	38.90	92.27	27.11	94.28
	GradNorm	23.73	93.97	42.81	87.26	55.62	81.85	38.15	87.73	40.08	87.70
	GradNorm+ReAct	12.95	97.74	26.41	94.85	38.44	91.70	29.55	93.78	26.84	94.52
	GradNorm+**BATS**	**10.01**	**98.23**	**18.87**	**96.42**	**32.45**	**92.78**	**24.79**	**95.28**	**21.53**	**95.68**
MNet	MSP	63.09	85.71	79.67	76.01	81.47	75.51	75.12	76.49	74.84	78.43
	MSP+ReAct	65.42	86.90	81.09	76.09	81.68	75.68	69.93	81.34	74.53	80.00
	MSP+**BATS**	49.77	90.60	70.75	80.66	74.66	78.45	57.61	85.61	63.20	83.83
	ODIN	45.61	91.33	63.03	83.44	70.01	80.85	52.45	85.61	57.78	85.31
	ODIN+ReAct	41.90	92.36	68.29	82.82	71.96	81.00	43.37	89.76	56.38	86.49
	ODIN+**BATS**	29.15	94.66	58.54	85.38	65.60	82.24	35.96	91.42	47.31	88.43
	Energy	49.52	91.10	63.06	84.42	69.24	81.42	58.16	84.88	60.00	85.46
	Energy+ReAct	37.08	93.41	53.13	86.04	54.15	83.31	42.45	89.42	46.70	88.05
	Energy+**BATS**	31.56	94.33	41.68	90.21	52.43	86.26	38.69	90.76	41.09	90.39
	GradNorm	33.70	92.46	42.15	89.65	56.56	83.93	34.95	90.99	41.84	89.26
	GradNorm+ReAct	25.85	95.03	38.94	91.42	52.94	86.74	18.85	95.75	34.15	92.24
	GradNorm+**BATS**	**21.51**	**95.89**	**30.97**	**93.19**	**46.94**	**88.08**	**17.71**	**95.97**	**29.28**	**93.28**

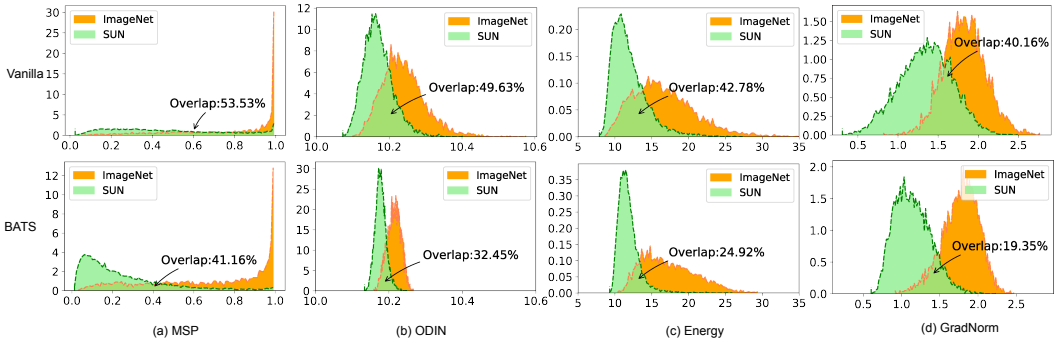


Figure 10: We illustrate different OOD scores for ID (ImageNet) and OOD (SUN) examples. "Vanilla" means the original OOD detection method, and "BATS" means applying our BATS to the detection method. BATS reduces the variance of the scores and reduces the overlap between the distribution of ID and OOD examples.

J The feature distribution on different channels

Fig. 11 illustrates the feature distribution on different channels of in-distribution examples (ImageNet) and the out-of-distribution examples on ResNet-50. These features are extracted by the last convolution block. We name the region where features are concentrated as the typical set of features. These regions receive more attention during training, and the model is more familiar with the features in these regions than those in extreme regions. For better visual presentation, we illustrate features before ReLU.

K Detection on natural adversarial examples

In our paper, we follow the settings of the existing research, choosing iNaturalist, Places, SUN and Textures as out-of-distribution datasets and choosing ImageNet-1k as the in-distribution dataset. Here we consider a much more challenging task: detecting natural adversarial examples [46]. Hendrycks et al. [46] introduce natural adversarial examples ImageNet-O, which are naturally occurring examples in the real world but significantly degrade the deep model performance. We use the ImageNet-O [46] which contains anomalies of unforeseen classes as the out-of-distribution examples. Fig. 12

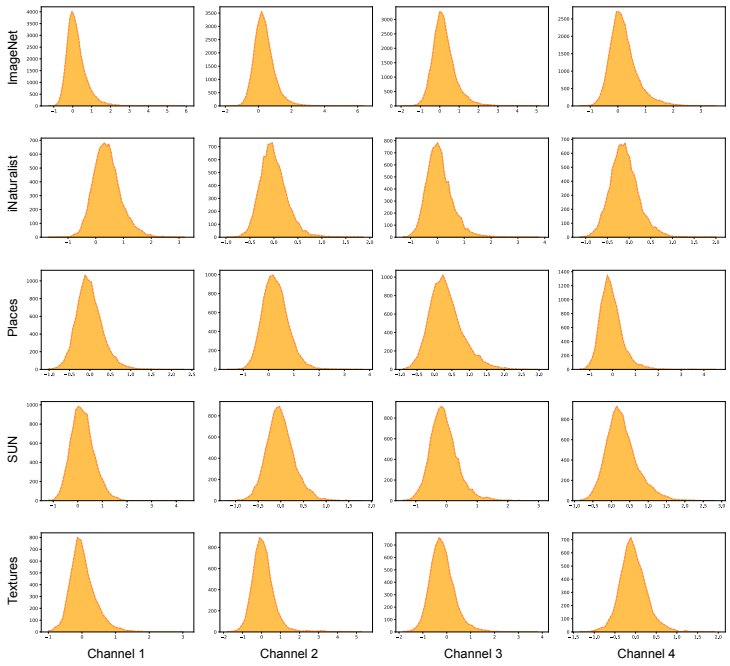


Figure 11: The feature distribution on different channels of in-distribution examples (ImageNet) and the out-of-distribution examples on ResNet-50. We randomly choose four channels of the features extracted by the penultimate layer.

shows that our method surpasses the existing methods by a large margin in FPR(1-α) with different significance levels. Our method significantly improve the AUROC from 56.68% to 64.48%.

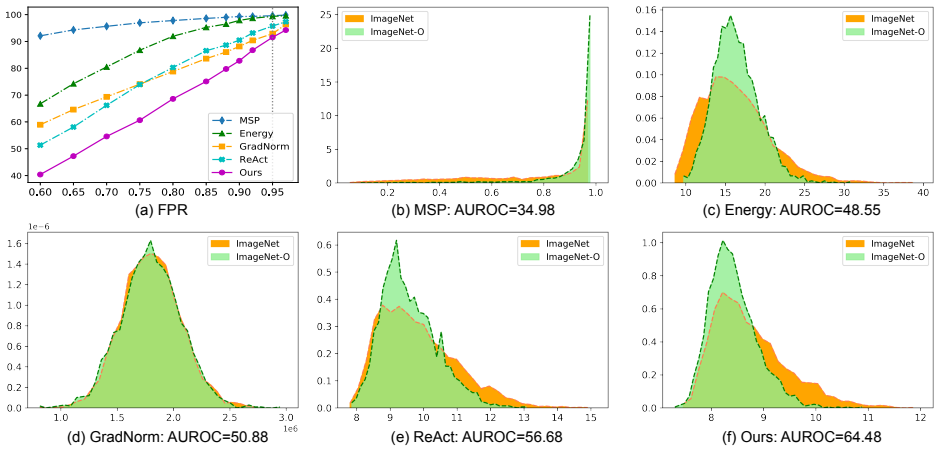


Figure 12: (a) The FPR(1-α) for different method on Imagenet (lower is better). The model is ResNet-50 and the OOD dataset is ImageNet-O. (b)-(f) The frequency histogram of the scores for ImageNet and ImageNet-O.

L Class activation mapping

In this section, we use the Smooth Grad-CAM++[57] to generate the heat map for different images. As shown in Fig. 13, the heat map of our rectified model aligns better with the objects in the image than that of the original model. We use the pre-trained ResNet-50 in PyTorch [45]. The heat map of our rectified model for Fig. 13(b) (the mud turtle) shows that the head of the turtle dominates the

decision while the original model pays more attention to the neck of the turtle. The rectified model takes more object-relevant parts into consideration, which may contribute to its slightly better test accuracy (in Appendix H).

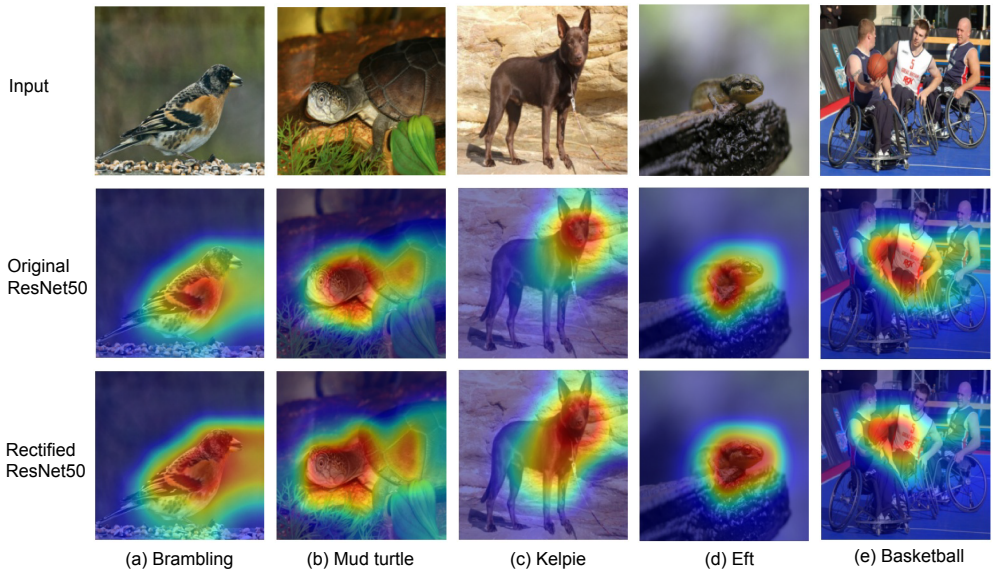


Figure 13: We draw the heat maps to explain which parts of the image dominate the model decision through Smooth Grad-CAM++[57]. The heat map of our rectified model for each image aligns better with the objects in the image than that of the original model.

M Additional Analysis for Performance Degradation Case in Tab.2

In Tab. 2, we show that the average performance of our **BATS** surpasses the baseline method Energy, but the performance degrades in the case using Tiny-Imagenet as the OOD dataset. We hypothesize that this performance degradation is due to the bias introduced by BATS. By truncating the features, BATS can reduce the variance of the in-distribution examples which benefits the estimation of the reject region but inherently cause some information loss which may reduce the performance of the pre-trained models.

To validate our hypothesis, we tune the bias-variance trade-off by the hyperparameter λ. As shown in Fig. 14, BATS can indeed reduce the variance of the OOD scores. With a proper λ, BATS can reduce the overlap between the ID and OOD examples and reduce the FPR95, while a small λ hinders the performance of OOD detection. For example, using larger $\lambda = 8$, BATS can achieve better FPR95 performance 15.10% on detecting Tiny-Imagenet using ResNet-18, which is 2.65% better than $\lambda = 3$ in our Tab. 2. For the practicability of our method, we set the same hyperparameter to test different OOD datasets, without adjusting for specific OOD datasets.

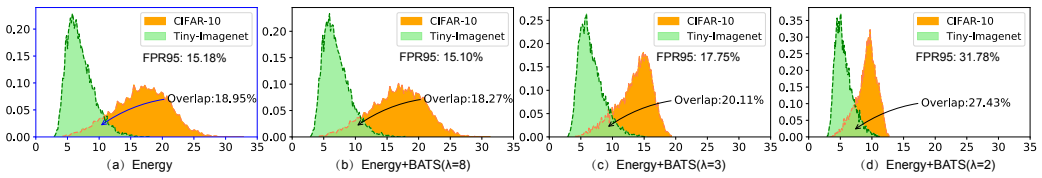


Figure 14: The distribution of the scores for ID (CIFAR-10) and OOD examples (Tiny-ImageNet) on ResNet-18. Choosing a proper λ, BATS can reduce the overlap between the ID and OOD examples and reduce the FPR95, while a small λ hinders the performance of OOD detection.

N The difference between BATS and ReAct

The similarity between our **BATS** and ReAct is that these methods are used to improve the performance of the existing OOD scores. As follows, we discuss the difference between **BATS** and ReAct from three aspects.

First, the motivation between our **BATS** and ReAct is different. ReAct hypothesizes that the mean activation of OOD data has significantly larger variations across units and is biased towards having sharp positive values, while the activation of the ID data is well-behaved with a near-constant mean and standard deviation. Thus, ReAct thinks that the truncation can rectify the activation of the OOD examples and preserve the activation of in-distribution data. However, as shown in Fig. 15, the mean activation of OOD data does not always have significantly larger variations than the ID data, which means this hypothesis does not always hold. The distribution of the deep features after batch normalization is consistent with the Gaussian distribution. Our **BATS** hypothesizes that deep models may be hard to model the extreme features but can provide reliable estimations on the typical features. This is because extreme features are exposed to the training process with a low probability. We propose to rectify the features into the typical set and calculate the OOD scores with the typical features.

Second, the mathematical analysis between our BATS and ReAct is different. ReAct theoretically analyze that if the OOD activations are more positively skewed, their operation reduces mean OOD activations more than ID activations. We analyze the benefit of **BATS** from the perspective of the bias-variance trade-off. BATS can reduce the variance of the deep features, which contributes to constraining the uncertainty of the test static $T(x; f)$ and improving the estimation accuracy of the reject region. Our method hopes to estimate the reject region better, and we do not assume whether OOD data is positively skewed.

Third, our method surpasses the ReAct in both the large-scale benchmark (ImageNet) and the small-scale benchmark (CIFAR).

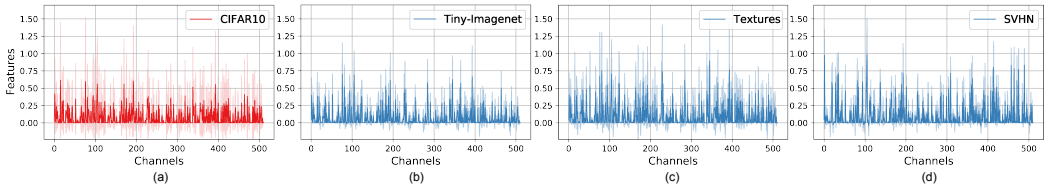


Figure 15: The distribution of the features of the in-distribution dataset (a) and out-of-distribution datasets (b-d) on different channels. We use the WideResNet-28-10 to extract the features. The mean and standard deviation are shown by the solid line and shaded area, respectively. Compared to other datasets, the mean value of the features in different channels of the Tiny-Imagenet is smaller and has a smaller standard deviation.

O Selecting features' typical set without assistance of BN

In our paper, we mainly analyze that rectifying the features in the typical set can improve the performance of the existing OOD scores. We provide a concise and effective method to select the typical set with the assistance of the BN layers and achieve a state-of-the-art performance among post-hoc methods on a suite of OOD detection benchmarks.

Here, we provide another method to select the features' typical set, which directly uses a set of training images to estimate the mean μ and the standard deviation σ of the features (extracted by the penultimate layer of the model) at each dimension. Then we rectify the features into the interval $[\mu - \lambda * \sigma, \mu + \lambda * \sigma]$ and use these typical features to calculate the OOD scores. We named this method as **T**ypical **F**eature **E**stimated **M**ethod (TFEM). This method does not require the BN layers in the model but needs to use a set of training images.

In Tab. 6, we compare the OOD detection performance of the OOD detection methods with and without our TFEM. In this experiment, we randomly choose 1500 images from the training dataset of the ImageNet. The λ is set to 1. The experiment is performed on the ImageNet benchmark. The

models are pre-trained ResNet-50 and ViT. ViT (Vision Transformer) [58] is a transformer-based image classification model which treats images as sequences of patches and does not have BN layers. We use the officially released ViT-B/16 model, which is pre-trained on ImageNet-21K and fine-tuned on ImageNet-1K. Rectifying the features into the typical set with TFEM can greatly improve the performance of the existing OOD detection methods both on the model with BN layers (ResNet-50) and the model without BN layers (ViT).

This experiment demonstrates the effectiveness of the typical features in OOD detection, which is consistent with the analysis in our paper. We believe there exists a method that can estimate the features' typical set better. In this paper, BATS has already established state-of-the-art performance on both the large-scale and small-scale OOD detection benchmarks.

Table 6: Using TFEM to select features' typical set. We use the pre-trained ResNet-50 and ViT-B/16 to detect the OOD examples. The best results are in bold.

Model	Method	iNaturalist		SUN		Places		Textures		Average	
		FPR95	AUROC	FPR95	AUROC	FPR95	AUROC	FPR95	AUROC	FPR95	AUROC
ViT	MSP	18.72	96.09	56.02	85.92	59.30	84.85	51.08	84.90	46.28	87.94
	MSP+**TFEM**	**7.12**	**98.34**	**42.62**	**90.36**	**48.71**	**88.64**	**41.29**	**88.48**	**34.94**	**91.46**
	ODIN	12.72	97.15	40.04	90.40	50.46	87.10	41.08	89.30	36.08	90.99
	ODIN+**TFEM**	**8.43**	**98.28**	**33.37**	**93.00**	**44.80**	**89.64**	**39.98**	**89.87**	**31.65**	**92.70**
	Energy	6.11	98.67	36.83	91.82	45.26	89.37	31.86	91.78	30.02	92.91
	Energy+**TFEM**	**3.59**	**99.08**	**30.27**	**93.70**	**41.53**	**90.51**	**31.13**	**92.13**	**26.63**	**93.86**
	GradNorm	7.66	98.43	48.36	85.28	66.38	72.85	48.79	80.76	42.80	84.33
	GradNorm+**TFEM**	**4.05**	**98.95**	**30.23**	**93.37**	**41.07**	**91.20**	**31.67**	**91.82**	**26.76**	**93.84**
ResNet50	MSP	51.44	88.17	72.04	79.95	74.34	78.84	**54.90**	78.69	63.18	81.41
	MSP+**TFEM**	**38.50**	**92.77**	**66.53**	**84.47**	**70.59**	**82.13**	58.40	**86.71**	**58.51**	**86.52**
	ODIN	41.07	91.32	64.63	84.71	68.36	81.95	50.55	85.77	56.15	85.94
	ODIN+**TFEM**	**28.40**	**94.67**	**52.34**	**89.47**	**62.13**	**85.14**	**37.27**	**92.35**	**45.04**	**90.41**
	Energy	46.65	91.32	61.96	84.88	67.97	82.21	56.06	84.88	58.16	85.82
	Energy+**TFEM**	**20.29**	**96.24**	**53.98**	**86.85**	**43.37**	**90.90**	**38.24**	**92.22**	**38.97**	**91.55**
	GradNorm	23.73	93.97	42.81	87.26	55.62	81.85	38.15	87.73	40.08	87.70
	GradNorm+**TFEM**	**11.88**	**97.83**	**26.24**	**95.00**	**40.46**	**90.77**	**25.05**	**94.85**	**25.91**	**94.61**

P Comparison between BATS and two latest detection methods

In this section, we compare our **BATS** with two latest OOD detection methods KNN [59] and ViM [60]. KNN is a nearest-neighbor-based OOD detection method, which computes the k-th nearest neighbor (KNN) distance between the embedding of test input and the embeddings of the training set to determine if the input is OOD or not. ViM combines the class-agnostic score from feature space and the In-Distribution class-dependent logits to calculate the OOD score. As shown in Tab. 7, our **BATS** outperforms the existing methods by a large margin. KNN explores and demonstrates the efficacy of the non-parametric nearest-neighbor distance for OOD detection, but its performance is worse than GradNorm and ReAct. ViM performs well on the OOD dataset Textures, but when using SUN as the OOD dataset, its performance is even worse than the simple baseline MSP.

Table 7: OOD detection performance comparison on ResNet-50 on the ImageNet benchmark. All methods are post hoc and can be directly used for pre-trained models. The best results are in Bold.

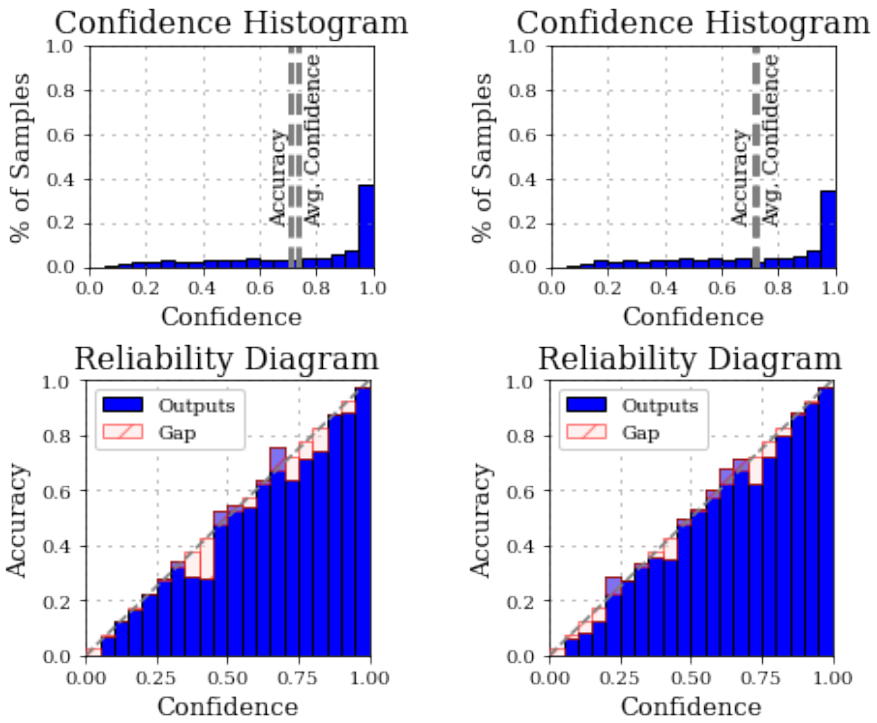
Method	iNaturalist		SUN		Places		Textures		Average	
	FPR95	AUROC	FPR95	AUROC	FPR95	AUROC	FPR95	AUROC	FPR95	AUROC
MSP [6]	51.44	88.17	72.04	79.95	74.34	78.84	54.90	78.69	63.18	81.41
ODIN [7]	41.07	91.32	64.63	84.71	68.36	81.95	50.55	85.77	56.15	85.94
Energy [5]	46.65	91.32	61.96	84.88	67.97	82.21	56.06	84.88	58.16	85.82
GradNorm [12]	23.73	93.97	42.81	87.26	55.62	81.85	38.15	87.73	40.08	87.70
ReAct [8]	17.77	96.70	25.15	94.34	34.64	91.92	51.31	88.83	32.22	92.95
KNN [59]	59.00	86.47	68.82	80.72	76.28	75.76	**11.77**	**97.07**	53.97	85.01
ViM [60]	77.34	86.46	90.71	73.80	89.64	72.15	16.63	96.37	68.58	82.20
BATS(Ours)	**12.57**	**97.67**	**22.62**	**95.33**	**34.34**	**91.83**	38.90	92.27	**27.11**	**94.28**

Q Benefits of BATS on calibration

The outputs of a classifier are often interpreted as the predictive confidence that this class was identified. Deep neural networks are often not calibrated which means that the confidence always

does not align with the misclassification rate. Expected Calibration Error (ECE) is a metric to measure the calibration of a classifier. For a perfectly calibrated classifier, the ECE value will be zero.

We use the reliability diagram to find out how well the classifier is calibrated in Fig. 16. The model's predictions are divided into bins based on the confidence value of the target class, here, we choose 20 bins. The confidence histogram shows how many test examples are in each bin. Two vertical lines represent the accuracy and average confidence, and the closer these two lines are, the better the model calibration is. BATS can improve the calibration of the pre-trained model and reduce the ECE of the pre-trained model from 3.56% to 2.12%.



(a) Original Model, ECE: 3.56% (b) BATS Model, ECE: 2.12%

Figure 16: We draw the reliability diagram and the confidence histogram of the pre-trained ResNet-50 (a) and the ResNet-50 with our BATS (b) on ImageNet.