

# What Matters for Out-of-Distribution Detectors using Pre-trained CNN?

Dong-Hee Kim<sup>\*1</sup><sup>a</sup>, Jaeyoon Lee<sup>\*1</sup><sup>b</sup> and Ki-Seok Chung<sup>1</sup><sup>c</sup>

<sup>1</sup>*Department of Electronic Engineering, Hanyang University, 222 Wangsimni-ro, Seoul, Korea*  
{dongheekim, zxc1421, kchung}@hanyang.ac.kr

Keywords: Out-of-Distribution Detection, Convolutional Neural Network

Abstract: In many real-world applications, a trained neural network classifier may have inputs that do not belong to any classes of the dataset used for training. Such inputs are called out-of-distribution (OOD) inputs. Obviously, OOD samples may cause the classifier to perform unreliably and inaccurately. Therefore, it is important to have the capability of distinguishing the OOD inputs from the in-distribution (ID) data. To improve the detection capability, quite a few methods using pre-trained convolutional neural networks (CNNs) with OOD samples have been proposed. Even though these methods show good performance in various applications, the OOD detection capabilities may vary depending on the implementation details and the methodology how to apply a set of detection methods. Thus, it is very important to choose both a good set of solutions and the methodology how to apply the set of solutions to maximize the effectiveness. In this paper, we carry out an extensive set of experiments to discuss various factors that may affect the OOD detection performance. Four different OOD detectors are tested with various implementation settings to find the configuration to achieve practically solid results.

## 1 Introduction

Since Hendrycks and Gimpel (Hendrycks and Gimpel, 2017) proposed a scheme for detecting out-of-distribution (OOD) samples, various deep learning methods have been suggested for detecting OOD samples. Thanks to the development, the reliability of the neural network classifier is improved. For instance, Oberdiek *et al.* (Oberdiek *et al.*, 2020) proposed an OOD detection method to solve a semantic segmentation task for a driver-centric view segmentation dataset, and Pacheco *et al.* (Pacheco *et al.*, 2020) proposed a method to pick out images that are not suitable for skin cancer classifiers.


Among many OOD detectors, detectors based on pre-trained convolutional neural networks (CNNs) such as ODIN (Liang *et al.*, 2018), Mahalanobis (Lee *et al.*, 2018b), and Gram (Sastry and Oore, 2020) have gained lots of attention. These methods have the advantages that the network model does not have to be re-trained. Namely, they do not require the full training dataset to conduct the end-to-end backpropagation. Furthermore, the pre-trained CNN-based detectors can maintain the in-distribution (ID) classifica-


tion accuracy. It is a considerable advantage because other previous works suffered from classification accuracy degradation (Lee *et al.*, 2018a; Hendrycks *et al.*, 2019a; Yu and Aizawa, 2019; Hsu *et al.*, 2020). Due to these merits, the pre-trained detectors are widely adopted in practical applications. Therefore, we will focus on the detectors based on pre-training in this paper.


Even though the OOD detectors perform well in general, the OOD detection capabilities may vary depending on the implementation details and the methodology how to apply a set of detection methods. OD-test (Shafaei *et al.*, 2019) addressed these issues with three different datasets. Nevertheless, more extensive study is necessary to have clues to

OOD	TNR at 95% TPR	AUROC
	Mahalanobis / G-ODIN	
SVHN	89.29 / <b>93.18</b>	98.02 / <b>98.69</b>
LSUN	<b>95.40</b> / 88.12	<b>98.99</b> / 97.50
TinyImageNet	<b>93.05</b> / 80.81	<b>98.66</b> / 96.05

Table 1: Mahalanobis detector based on pre-trained CNN (Lee *et al.*, 2018b) outperforms Generalized-ODIN (Hsu *et al.*, 2020) for LSUN and TinyImageNet (in-distribution dataset: CIFAR-10). All values are percentages averaged over five times, and the best results are indicated in bold.

<sup>a</sup> <https://orcid.org/0000-0003-4787-8016>

<sup>b</sup> <https://orcid.org/0000-0003-1350-0357>

<sup>c</sup> <https://orcid.org/0000-0002-2908-8443>

achieve stable and reliable OOD detection capability. For instance, our experimental results using open-source implementations show that an old-fashioned pre-trained detector outperforms one of the latest approaches for detecting OOD samples. Table 1 shows the performance comparison results where the Mahalanobis detector based on pre-trained CNN (Lee et al., 2018b) outperforms Generalized-ODIN (Hsu et al., 2020) for LSUN and TinyImageNet. The details will be elaborated further in Section 3.5.

Our paper aims to figure out what would influence the performance of the pre-trained OOD detectors most. This paper is not supposed to compare OOD detectors with each other. In that respect, we carry out an extensive set of experiments with various implementation settings to figure out a set of key factors to affect the OOD detection. From this perspective, our contribution can be summarized as (1) implementing four OOD detectors that are based on pre-trained CNNs with various configuration options, (2) verifying the influences of various implementation settings by carrying out an extensive set of OOD detection tests, and (3) suggesting competitive sets of options for practical applications.

The remainder of this paper is organized as follows. In Section 2, we describe our experimental setting and explain the evaluated OOD detectors and the evaluation metrics. In Section 3, we evaluate the performance of the OOD detectors under various circumstances and analyze their OOD detection performance. Section 4 concludes this paper.

Our code is available in <https://github.com/LJY-HY/What-matters-for-ODD-detector.git>.

## 2 Background

**Performance evaluation setting** In this paper, we consider only the OOD detection methods that are based on pre-trained CNN-based models. Specifically, we consider the detectors which do not change the network’s parameters through additional training. Since the model’s inference cost for classification is closely related to the model’s structure, both the inference cost and the classification capability will remain unchanged even after the detection methods are applied.

We evaluate the following OOD detectors with pre-trained CNNs:

1. **Baseline** (Hendrycks and Gimpel, 2017) is the basic approach for detecting the OOD samples with maximum softmax probability (MSP) as an anomaly score. The baseline detector regards a sample with a high MSP score as an ID and does

one with a low score as an OOD. Thus, Baseline does not need any additional calculation except for the one for the original inference.

2. **ODIN** (Liang et al., 2018) achieves an impressive improvement of the detection performance by tweaking the neural network’s inputs and outputs. First, a small perturbation  $\epsilon$  to the input image to figure out ID and OOD is applied. Then, the logits before the softmax layer are scaled by  $1/T$  where  $T$  and  $\epsilon$  are hyperparameters determined by the same process as the original paper; making candidate lists of hyperparameters and carrying out a grid search. It means, to find out the optimal set of the hyperparameters, we need a few ID and OOD samples.
3. **Mahalanobis** (Lee et al., 2018b) measures the Mahalanobis distance between an ID data and an input sample for each feature. Here, features denote the outputs of convolution layers in a CNN, described as the input image representations. The measured distance becomes the anomaly score by regarding samples within a short distance as ID and ones outside a certain distance as OOD. For more accurate analysis, the authors proposed an input pre-processing and a feature ensemble. The input pre-processing at Mahalanobis is the same as ODIN, but ODIN’s perturbation is based on the target label, while Mahalanobis’ perturbation is based on the measured Mahalanobis distance. Accordingly, we need perturbation magnitude  $\epsilon$  in Mahalanobis as well. We can calculate the Mahalanobis distance of each feature in terms of the feature ensemble, then train a logistic regression detector on these distances.
4. **Gram** (Sastry and Oore, 2020) utilizes a matrix called Gram matrix that is composed of both the model’s prediction and the intermediate values. The Gram matrix contains information on the activations at the individual channels and summarizes the pairwise interactions between the channels. The subsequent scoring method is similar to Mahalanobis’. However, the Gram detector differs from Mahalanobis in that a higher-order correlation between features is considered, and no hyperparameter needs to be adjusted.

We measure the OOD detection performance using the following metrics: (1) **TNR at 95%**; TNR denotes the true negative rate (the ratio of OOD samples which are classified as OOD) when the true positive rate (TPR, the ratio of ID samples which are classified as ID) is 95%. TNR is calculated as  $TN/(TN+FP)$ . TPR is measured as  $TP/(TP+FN)$  where TP, TN, FP, and FN denote True Positive, True Negative, False

Positive and False Negative, respectively. (2) **AU-ROC** that implies the area under an ROC curve where the ROC curve is a plot of TPR on the y-axis and FPR on the x-axis, and (3) **Detection Accuracy** that is calculated as the total number of the true positive samples and the true negative samples divided by the total number of samples. The threshold value for classifying the true positive and true negative samples is determined when the detection accuracy reaches the maximum value among all the possible thresholds. All metrics indicate a better OOD detection performance when they get as close as 1.

### 3 Experiments

This paper aims to investigate the OOD detection performance of a pre-trained model from diverse perspectives. For the experiments, seven perspectives are selected and each method is evaluated for the OOD detection performance by varying their settings.

When evaluating the performance of the detectors, widely used datasets such as CIFAR-10 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), LSUN (Yu et al., 2015), TinyImageNet (Deng et al., 2009), and CIFAR-100 (Krizhevsky et al., 2009) are used. CIFAR-10 and CIFAR-100 are used as the ID datasets, and the others are used as the OOD datasets. Also, CIFAR-10 and CIFAR-100 are used as the OOD dataset to each other.

Seven categories that supposedly affect the OOD detection capability are selected as follows: *semantics-preserved OOD samples* (Section 3.1), *hyperparameters for training pre-trained CNN* (Section 3.2), *hyperparameters for network architecture* (Section 3.3), *classification refinement tricks* (Section 3.4), *designing auxiliary dataset* (Section 3.5), *the number of training samples* (Section 3.6), and *pre-trained by supervised contrastive learning* (Section 3.7).

#### 3.1 Detection of Semantics-Preserved OOD Samples

**Study Description** This section closely analyzes the OOD detection capabilities of the compared detectors depending on the type of dataset. The CNN-based OOD detectors extract semantics through CNN first and then distinguish whether the data is ID or not by leveraging the extracted semantics. Therefore, we may expect that a well-preserved semantics would be an advantage for better OOD detection. To verify whether such expectation turns out to be true, we set up experiment environments to test two different sets

of datasets: conventionally used datasets and semantically preserved datasets. For conventional datasets, LSUN and TinyImageNet are used. For semantically preserved ones, LSUN FIX and TinyImageNet FIX which were recently suggested by Tack *et al.* (Tack et al., 2020) are used. In order to preserve semantics, a resizing transformation is applied to the original picture. More details about semantic preservation are described in the appendix.

**Discussion** On the contrary to our expectation, as shown in Table 2, the detection performances on LSUN and TinyImageNet are better than those on LSUN FIX and TinyImageNet FIX in all cases. Therefore, it may be claimed that the existing OOD detecting methods focus more on data statistics such as smoothness of the input than on the true meaning of the data. The Mahalanobis detector’s AUROC for LSUN drops from 99.63% to 87.91% when CIFAR-10 is ID. When CIFAR-100 is ID, the performance drops to a larger extent, 98.63% to 70.61%. This trend becomes clearer as the detector’s performance for the conventional benchmark improves. However, the amount of the performance drop differs depending on both each combination of ID and OOD datasets and the OOD detection methods. Therefore, all subsequent experiments in this paper conduct performance evaluation on both types of the OOD datasets.

**Suggestion** From the investigation of this section, we claim that it is desirable to expand the benchmark by including semantically preserved datasets. Thereby, the performance of the OOD detector can be evaluated more comprehensively with the expanded benchmark.

#### 3.2 Hyperparameters for Training CNN

**Study Description** Unlike the methods that carry out structural transformation (Hsu et al., 2020; Yu and Aizawa, 2019; Hendrycks et al., 2019a), the detectors applied to the pre-trained CNN network maintain the base network’s classification accuracy. To investigate how much the OOD detection performance will be affected by the training environment, an extensive set of experimental environments is organized. We want to investigate how much the OOD detection performance depends on the model’s training method by varying the hyperparameter configuration such as batch size, optimizer, scheduler, weight decay and epoch that are originally determined to boost the classification accuracy. For each experiment, a network is trained by varying a single hyperparameter while keeping the others set to the baseline setting. More details of the baseline setting are described in the appendix.

ID	OOD	TNR at 95% TPR				AUROC				Detection Accuracy				
		Baseline / ODIN / Mahalanobis / Gram				Baseline / ODIN / Mahalanobis / Gram				Baseline / ODIN / Mahalanobis / Gram				
CIFAR-10	SVHN	56.76 / 69.08 / <b>99.46</b> / 97.39	93.36 / 93.82 / <b>99.64</b> / 99.43	88.43 / 88.28 / <b>98.05</b> / 96.62										
	LSUN	58.42 / 81.98 / <b>98.61</b> / 95.27	93.64 / 95.35 / 99.63 / <b>99.85</b>	88.25 / 89.99 / 97.53 / <b>98.55</b>										
	TinyImageNet	49.74 / 70.99 / 97.03 / <b>98.63</b>	91.02 / 91.84 / 99.35 / <b>99.70</b>	85.46 / 86.03 / 96.36 / <b>97.59</b>										
	LSUN FIX	46.23 / <b>58.84</b> / 40.93 / 32.81	<b>90.24</b> / 90.20 / 87.91 / 84.64	<b>84.78</b> / 84.43 / 80.89 / 77.24										
	TinyImageNet FIX	46.77 / <b>56.99</b> / 48.91 / 39.36	<b>89.97</b> / 88.99 / 89.13 / 85.12	<b>85.46</b> / 83.47 / 81.90 / 77.26										
	CIFAR-100	41.77 / <b>49.48</b> / 47.67 / 33.06	88.00 / 86.93 / <b>88.27</b> / 80.95	<b>82.68</b> / 81.50 / 80.93 / 73.52										
CIFAR-100	SVHN	27.10 / 63.74 / <b>93.43</b> / 80.01	83.43 / 94.79 / <b>98.12</b> / 96.01	76.14 / 89.70 / <b>95.24</b> / 89.48										
	LSUN	28.48 / 58.79 / 95.60 / <b>97.92</b>	82.25 / 92.15 / 98.63 / <b>99.47</b>	74.81 / 84.60 / 95.38 / <b>96.92</b>										
	TinyImageNet	30.15 / 61.17 / 90.21 / <b>96.13</b>	82.73 / 92.81 / 98.01 / <b>99.10</b>	74.94 / 85.35 / 92.98 / <b>95.77</b>										
	LSUN FIX	13.07 / <b>15.57</b> / 13.48 / 10.74	73.14 / <b>73.97</b> / 67.65 / 64.62	68.45 / <b>69.07</b> / 63.37 / 60.95										
	TinyImageNet FIX	22.63 / <b>24.97</b> / 12.69 / 20.54	78.58 / <b>78.73</b> / 70.61 / 74.19	72.28 / <b>72.69</b> / 65.93 / 68.34										
	CIFAR-10	20.60 / <b>20.77</b> / 8.17 / 12.80	78.78 / <b>79.45</b> / 62.63 / 68.79	72.16 / <b>72.93</b> / 59.74 / 59.39										
SUBCIFAR-10	SVHN	9.08 / 53.77 / <b>94.91</b> / 90.73	71.15 / 88.47 / <b>98.32</b> / 97.77	68.17 / 81.27 / <b>95.15</b> / 93.00										
	LSUN	18.94 / 49.82 / 89.98 / <b>94.19</b>	78.11 / 90.06 / 97.09 / <b>98.56</b>	72.01 / 82.26 / 93.13 / <b>94.91</b>										
	TinyImageNet	15.14 / 38.56 / 81.86 / <b>89.03</b>	72.71 / 85.25 / 95.53 / <b>97.23</b>	67.18 / 77.26 / 89.72 / <b>92.42</b>										
	LSUN FIX	18.35 / <b>27.14</b> / 5.12 / 12.89	78.10 / <b>82.66</b> / 52.29 / 68.05	72.06 / <b>75.43</b> / 52.74 / 64.23										
	TinyImageNet FIX	18.36 / <b>28.02</b> / 16.07 / 22.78	77.56 / <b>81.43</b> / 63.54 / 70.89	71.55 / <b>74.20</b> / 60.17 / 65.80										
	CIFAR-100	15.85 / <b>22.29</b> / 15.76 / 19.23	75.65 / <b>77.74</b> / 62.29 / 67.27	70.07 / <b>71.10</b> / 59.39 / 63.29										

Table 2: CIFAR-10 pre-trained network’s OOD detection performance on the expanded benchmark. The bold-faced results show the best performance among the four detectors. Each class in SUBCIFAR-10 is composed of randomly picked 500 images from CIFAR-10.

**Discussion** Our main goal in this section is to figure out how much the model’s OOD detection performance will be affected by each hyperparameter. The following five types of hyperparameters are taken into account.

(a) *batch size* For Baseline and ODIN, the batch size of 128 shows the best OOD detection performance. For Mahalanobis, the best batch size turns out to be 256 for the OOD detection performance. Even if the most suitable batch size differs for different detectors, it seems that there is little difference in the overall performance. However, a significant performance degradation occurs depending on the type of the OOD dataset. The performance of the ODIN detector on LSUN FIX is degraded considerably when the batch size changes from 128 to 256 when CIFAR-100 is ID. The performance of Mahalanobis also fluctuates substantially depending on the batch size when CIFAR-100 is ID and TinyImagenet is OOD.

(b) *optimizer* In Figure 1, both the SGD and Nesterov (Sutskever et al., 2013) optimizers show a superior performance than Adam (Kingma and Ba, 2015) or AdamW (Loshchilov and Hutter, 2019) except for one case. The ODIN detector shows a better performance when the optimizer is either Adam or AdamW than when the optimizer is either SGD or Nesterov. Specifically, it works well when CIFAR-100 is ID and the OODs are semantically preserved datasets such as LSUN FIX and TinyImagenet FIX.

(c) *learning rate scheduler* In Figure 1, *cosine annealing* (Loshchilov and Hutter, 2017) shows a better performance than the other two schedulers for

Baseline and ODIN. For Mahalanobis and Gram, *cosine annealing with warm up* (Loshchilov and Hutter, 2017) seems to be a better choice. In case of the CIFAR-100 ID, the optimal choice for Baseline and ODIN is not *cosine annealing* but *MultistepLR*. In detail, the best learning rate scheduler for each detector improves the AUROC performance for every benchmark. In particular, the best scheduler for Mahalanobis or Gram tends to significantly improve the detection performance on the semantically preserved dataset.

(d) *weight decay* When CIFAR-10 is ID, the OOD detection performance of the detectors varies significantly depending on the weight decay. While the optimal weight decay for Mahalanobis and Gram is  $5e-4$ , that for Baseline is  $1e-4$ . Surprisingly, ODIN improves its OOD detection performance as the weight decay value gets smaller. The best weight decay for the CIFAR-100 ID is also  $5e-4$ .

(e) *epoch* Though the OOD detection performance gets better until epoch 300, performance drops are observed in several cases as epoch reaches 400. For example, when CIFAR-10 is ID, ODIN’s AUROC drops on LSUN FIX when epoch changes from 300 to 400. AUROC of the Mahalanobis and Gram detectors falls when epoch changes from 300 to 400 to detect the CIFAR-100 OOD. Similar tendencies are observed when CIFAR-100 is ID.

**Suggestion** One of the key observations is that the optimal configuration for OOD detection performance and classification accuracy frequently mismatch. The following recommendations can be made: for detect-

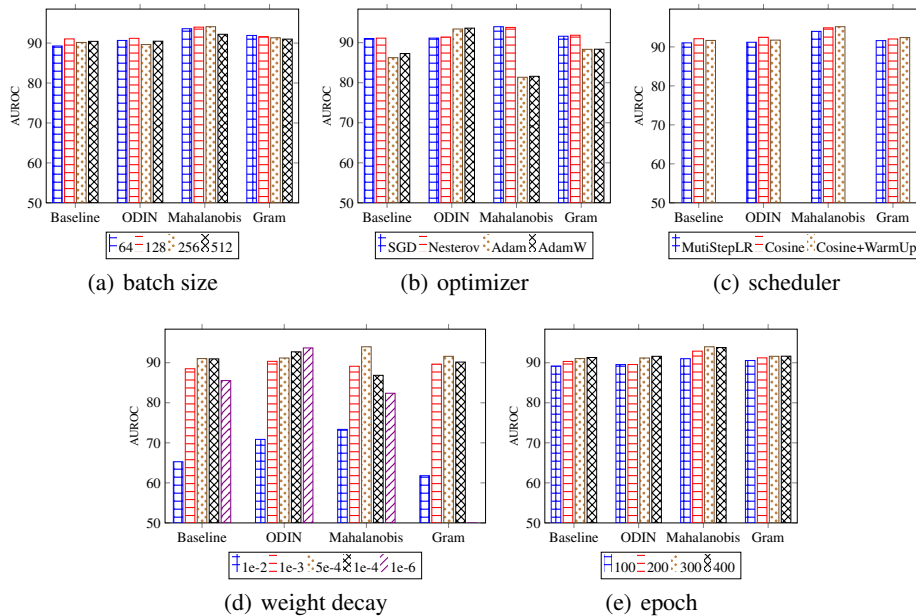


Figure 1: CIFAR-10 AUROC performances according to the hyperparameters

ing OOD samples, the best batch size is 128, and for the optimizer, SGD or Nesterov over Adam or AdamW is preferred. In case of the scheduler, *MultiStepLR* is recommended for Baseline and ODIN. On the other hand, *cosine annealing with warmup* is recommended for Mahalanobis and Gram. Weight decay around  $5e-4$  is recommended for training. However, for some detector such as ODIN, this may not be the case. Nevertheless, in most cases, weight decay around  $5e-4$  yields the best performance. Lastly, epoch between 300 and 400 is recommended for training.

### 3.3 Hyperparameters for Network Architecture

**Study Description** The previous section considered a set of hyperparameters for training a network. In this section, the structural hyperparameters of a network are taken into account. First, the depth of a network is considered. It is commonly regarded that as the number of layers increases, the network’s classification performance gets better. To observe the correlation between the number of layers and the OOD detection capability, ResNets (He et al., 2016) with 18, 34, 50 and 101 layers are tested. Another structural hyperparameter considered in this section is the activation function. GeLU (Hendrycks and Gimpel, 2016) is used in some of the earlier works (Hendrycks and Gimpel, 2017). Also, ReLU (Nair and Hinton,

2010) is one of the most widely used activation functions. In addition to these two activation functions, we conduct experiments with some of the other popular activation functions such as leaky ReLU (Maas et al., 2013) and SiLU (Ramachandran et al., 2018).

**Discussion** From the experiments that we carried out, the deeper network does not guarantee a better OOD detection performance contrary to the classification accuracy. Rather, it is notable that for Baseline and ODIN, the network with the fewest layers performs best. Moreover, as the depth of the model increases, both the memory usage and the computation time increase proportionally. Gram’s excessive memory usage also hinders completing experiments on ResNet-50 and ResNet-101.

In case of activation functions, Baseline with ReLU and ODIN with SiLU perform slightly better than the others. However, there seems to be no clear winner or loser among the activation functions. On the other hand, Mahalanobis and Gram have some combinations that show significantly disappointing performance. Especially, the Gram detector with all the activation functions except for ReLU suffers from a 50% AUROC loss. Both Mahalanobis and Gram with all the activation functions except for ReLU spit out a feature that contains negative values to seemingly mark the anomaly score to make it difficult to distinguish OOD from ID.

**Suggestion** Even though a deeper model is preferred for the classification tasks, a shallow model is

recommended when detecting OOD samples. Also, a light-weight model will allow more memory to be available for the OOD detection. In addition, from the experimental results, we learn that using ReLU as the activation function is the best choice.

### 3.4 Classification Refinement Tricks

**Study Description** Many refinement tricks have been proposed to improve the classification accuracy. Label smoothing (Szegedy et al., 2016), mixup (Zhang et al., 2018), and knowledge distillation (Hinton et al., 2015) are some of those tricks. These tricks are known to enhance model’s generalization, calibration, and classification accuracy (Müller et al., 2019; Zhang et al., 2021; Phuong and Lampert, 2019). In other words, they provide clearer decision boundaries among classes of the ID samples. As the OOD detection using pre-trained CNN is aimed at improving the OOD detection performance while largely maintaining the classification accuracy, we investigate whether these refinement tricks should help to find a clear decision boundary between the ID and OOD samples.

**Discussion** Each refinement trick has pros and cons. For example, a trained model by knowledge distillation leads to some performance improvement with the Mahalanobis detector. On the other hand, label smoothing reveals some weakness with Baseline and ODIN. Especially, when LSUN FIX is used as the OOD samples, ODIN’s AUROC drops from 90.2% to 76.5% with the CIFAR-10 ID and from 73.36% to 60.97% with the CIFAR-100 ID. Mixup achieves the highest AUROC with the Gram detector, but overall, it is not dominantly good at the OOD detection task. Interestingly, the mixup method is the refinement trick that records the best ID classification accuracy.

**Suggestion** From the experimental result, we suggest that knowledge distillation should be employed if you want to improve both the classification accuracy and the OOD detection performance. The other refinement options have cases to show pros and cons.

### 3.5 Designing Auxiliary Dataset

**Study Description** For ODIN and Mahalanobis, a few OOD samples called *auxiliary* dataset are necessary. The auxiliary dataset is used when both selecting  $T$ ,  $\epsilon$  (Liang et al., 2018; Lee et al., 2018b) and tuning logistic regressors (Lee et al., 2018b). Selection and tuning are conducted in such a way that the ID set should be best distinguished from the auxiliary dataset. Liang *et al*(Liang et al., 2018) and Lee *et al*(Lee et al., 2018b) used 1,000 samples of

ID and OOD each for finding the best  $T$ ,  $\epsilon$  and the logistic regressor. However, the auxiliary dataset may not be available, and the tuning strategy has to be adjusted even under such circumstance. In this section, three different tuning strategies that can be applied without accessing the auxiliary dataset are discussed. *Adversarial*, proposed by Lee *et al*(Lee et al., 2018b), regards adversarial samples generated by FGSM (Goodfellow et al., 2015) as OOD. Meanwhile, rotation and permutation are known to be the strongest transformation methods among various shifting transformation methods (Tack et al., 2020; Chen et al., 2020). Therefore, *Rotation* and *Permutation* strategies regard rotated or permuted ID dataset as OOD. We apply these three tuning strategies to the ODIN and Mahalanobis detectors instead of their original tuning strategies.

**Discussion** For the ODIN detector, *Adversarial* works well with conventional benchmarks. Meanwhile, *Rotation* achieves the best AUROC performance for semantically preserved datasets. Sometimes, *Rotation* works even better than the ODIN’s original tuning strategy. Even though *Adversarial* does not achieve the best performance, its performance does not fall far behind *Rotation* either.

For the Mahalanobis detector, *Adversarial* is not the best for some cases, but overall, it shows decent performance for conventional benchmarks. For semantically preserved datasets, *Rotation* shows the best performance with CIFAR-10 as ID. With CIFAR-100 as ID, though *Permutation* is the best tuning strategy with the semantically preserved datasets, all three strategies show results that are too bad to be used in practical situations.

**Suggestion** As the tuning method for ODIN and Mahalanobis, it can be suggested that *Rotation* should be used as an OOD auxiliary set.

### 3.6 The Size of Samples in the ID Dataset

**Study Description** In general, the OOD detection capability as well as the classification accuracy is largely affected by three factors: the distribution difference, the size of samples per class, and the number of classes. In this section, we will figure out how the size of the training dataset (both the number of samples and the number of classes) affects the detector’s performance. In this experiment, CIFAR-10 and CIFAR-100 are used as one is ID and the other is OOD, and vice versa. Among three factors that affect performance, the distribution difference is not as significant as the other two because both datasets are originally selected from the same dataset (80 mil-

lion tiny images (Birhane and Prabhu, 2021)). To verify the effect of the other two factors, we construct a new dataset that is made up of a partial set of CIFAR-10, called *SUBCIFAR-10*. Each class in SUBCIFAR-10 is composed of randomly picked 500 images from CIFAR-10. With a model trained by SUBCIFAR-10, we analyze the effect of the number of classes and the size of samples per class on the OOD detection performance.

**Discussion** From the results, it turns out that the size of samples per class is much more important than the number of classes for the OOD detection accuracy. In Table 2, the effect of the number of samples in each class can be analyzed by comparing the result on CIFAR-10 and that on CIFAR-100. When SUBCIFAR-10 is used as the ID dataset, there is an AUROC performance drop by up to 20% for all detectors compared to the performance when the full CIFAR-10 is used as the ID set. Comparing the case where CIFAR-100 is used as the ID dataset with the case where the SUBCIFAR-10 as the ID dataset, the performance of the case with SUBCIFAR-10 falls behind almost every case. This implies that the number of classes does not influence the detection performance as much as the size of samples does for the OOD detection.

**Suggestion** Collecting more training data is an effective way to improve the OOD detection capability regardless of all detectors. If it is impossible, increasing the size of the training set through adding classes is a decent alternative to improve the performance.

### 3.7 Pre-trained by Supervised Contrastive Learning

**Study Description** Hendrycks *et al.*(Hendrycks *et al.*, 2019b) claimed that self-supervised learning could improve the OOD detection capability, and similar approaches were proposed by Tack *et al.*(Tack *et al.*, 2020) and Winkens *et al.*(Winkens *et al.*, 2020). In conjunction with contrastive learning, these studies successfully enhanced the OOD detection performance. In this section, to verify the claim, the OOD detection performance of the model pre-trained with contrastive learning is evaluated. For a fair comparison, we select *supervised contrastive learning* (Supcon) (Khosla *et al.*, 2020), trained under supervised environment.

**Discussion** Supcon achieves an impressive performance improvement in Baseline and ODIN. What we would like to emphasize in this experiment is that Supcon is surprisingly talented at picking out OOD samples from LSUN FIX, TinyImageNet FIX, and CIFAR-100 when CIFAR-10 as ID. In contrast, none

of the others is capable of doing that. Nevertheless, with the Gram detector, Supcon does not generate an excellent result. An experiment with CIFAR-10 as ID, LSUN FIX as OOD, AUROC drops from 84.64% to 72.6%.

Supervised contrastive learning attempts to learn the representation where samples of the same label are located to be closer to each other and samples of different labels to be farther away. Therefore, when an OOD sample is fed to the model, its representation should not be similar to that of any ID class. This phenomenon makes Supcon show great results with Baseline and ODIN. On the other hand, Mahalanobis and Gram, which already make use of features between samples for the OOD detection, are not so benefited by contrastive learning.

**Suggestion** Supervised contrastive learning may be an effective alternative with a large number of data. However, using cross-entropy loss may be a safer choice for Mahalanobis and Gram.

## 4 Conclusion

For practical applications, distinguishing out-of-distribution (OOD) samples from in-distribution (ID) samples is essential for making a neural network reliable. In this paper, we tackled the lack of diversity of the previous works by carrying out an extensive set of experiments to discover the unrevealed aspects of the OOD detection of pre-trained CNN-based models. By conducting the OOD detection tests over 7,000 times, we verified some of the importance factors that were not mentioned in the previous literature. From the extensive experimental results, various novel suggestions were made. One of the interesting observations is that pre-trained CNN-based detectors are vulnerable to semantics-preserved OOD samples, meaning that they do not focus on what images really mean. We hope this study may be a solid guide for the future OOD detection studies.

## Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2021-0-00131, Development of Intelligent Edge Computing Semiconductor For Lightweight Manufacturing Inspection Equipment)

## REFERENCES

- Birhane, A. and Prabhu, V. U. (2021). Large image datasets: A pyrrhic win for computer vision? In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1537–1547.
- Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. E. (2020). Big self-supervised models are strong semi-supervised learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22243–22255. Curran Associates, Inc.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.
- Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.
- Hendrycks, D. and Gimpel, K. (2017). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *Proceedings of International Conference on Learning Representations*.
- Hendrycks, D., Mazeika, M., and Dietterich, T. (2019a). Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*.
- Hendrycks, D., Mazeika, M., Kadavath, S., and Song, D. (2019b). Using self-supervised learning can improve model robustness and uncertainty. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.
- Hsu, Y.-C., Shen, Y., Jin, H., and Kira, Z. (2020). Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10951–10960.
- Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. (2020). Supervised contrastive learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc.
- Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Krizhevsky, A. et al. (2009). Learning multiple layers of features from tiny images.
- Lee, K., Lee, H., Lee, K., and Shin, J. (2018a). Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *International Conference on Learning Representations*.
- Lee, K., Lee, K., Lee, H., and Shin, J. (2018b). A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7167–7177.
- Liang, S., Li, Y., and Srikant, R. (2018). Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*.
- Loshchilov, I. and Hutter, F. (2017). SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.
- Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *International Conference on Learning Representations*.
- Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer.
- Müller, R., Kornblith, S., and Hinton, G. E. (2019). When does label smoothing help? In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *icml*.
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.
- Oberdiek, P., Rottmann, M., and Fink, G. A. (2020). Detection and retrieval of out-of-distribution objects in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.
- Pacheco, A. G. C., Sastry, C. S., Trappenberg, T., Oore, S., and Krohling, R. A. (2020). On out-of-distribution detection algorithms with deep neural skin cancer classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.
- Phuong, M. and Lampert, C. (2019). Towards understanding knowledge distillation. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*,



volume 97 of *Proceedings of Machine Learning Research*, pages 5142–5151. PMLR.

- Ramachandran, P., Zoph, B., and Le, Q. V. (2018). Searching for activation functions.
- Sastry, C. S. and Oore, S. (2020). Detecting out-of-distribution examples with gram matrices. In *International Conference on Machine Learning*, pages 8491–8501. PMLR.
- Shafaei, A., Schmidt, M., and Little, J. J. (2019). A less biased evaluation of out-of-distribution sample detectors. In *30th British Machine Vision Conference 2019, BMVC 2019, Cardiff, UK, September 9-12, 2019*, page 3. BMVA Press.
- Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA. PMLR.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Tack, J., Mo, S., Jeong, J., and Shin, J. (2020). Csi: Novelty detection via contrastive learning on distributionally shifted instances. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11839–11852. Curran Associates, Inc.
- Winkens, J., Bunel, R., Roy, A. G., Stanforth, R., Natarajan, V., Ledsam, J. R., MacWilliams, P., Kohli, P., Karthikesalingam, A., Kohl, S., et al. (2020). Contrastive training for improved out-of-distribution detection. *arXiv preprint arXiv:2007.05566*.
- Yu, F., Zhang, Y., Song, S., Seff, A., and Xiao, J. (2015). Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*.
- Yu, Q. and Aizawa, K. (2019). Unsupervised out-of-distribution detection by maximum classifier discrepancy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9518–9526.
- Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.
- Zhang, L., Deng, Z., Kawaguchi, K., Ghorbani, A., and Zou, J. (2021). How does mixup help with robustness and generalization? In *International Conference on Learning Representations*.

## APPENDIX

### Detailed Explanation of Compared Detectors with Pre-trained CNNs

**ODIN** makes use of perturbation noise generated by the FGSM method that back-propagates the gradients of values after softmax,  $S_{\hat{y}}$ . However, it is different in the sense that only its sign is used and then  $-\epsilon$  is multiplied. The method to make perturbed inputs is as follows:

$$\tilde{x} = x - \epsilon \text{sign}(-\nabla_x S_{\hat{y}}(x; T)) \quad (1)$$

After the perturbed input  $\tilde{x}$  is passed through the network, temperature scaling is applied by dividing logits  $f(x)$  by temperature  $T$ . Temperature scaling for input  $x$  is as follows:

$$S(x; T) = \frac{\exp(f(x)/T)}{\sum_{j=1}^C \exp(f_j(x)/T)}, \quad (2)$$

where  $C$  is the number of classes and  $T$  denotes temperature for scaling.

In our experiments, candidates for  $T$  and  $\epsilon$  are [1, 10, 100, 1000] and [0, 0.0005, 0.001, 0.0014, 0.002, 0.0024, 0.005, 0.01, 0.05, 0.1, 0.2], respectively.

**Mahalanobis** defines the Mahalanobis scores based on the Mahalanobis distance. To measure the Mahalanobis distance, empirical class mean  $\hat{\mu}_c$  and covariance  $\hat{\Sigma}$  of training samples  $\{(x_1, y_1), \dots, (x_N, y_N)\}$  are calculated for every layer as:

$$\begin{aligned} \hat{\mu}_c &= \frac{1}{N_c} \sum f(x_i), \\ \hat{\Sigma} &= \frac{1}{N} \sum_c \sum_{i: y_i=c} (f(x_i) - \hat{\mu}_c)(f(x_i) - \hat{\mu}_c)^T, \end{aligned} \quad (3)$$

where  $N_c$  is the number of training samples with label  $c$ .

After calculating the class mean and covariance, Mahalanobis defines confidence score  $M(x)$  at every layer using the Mahalanobis distance between test sample  $x$  and the closest class-conditional Gaussian distribution as:

$$M(x) = \max_c -(f(x) - \hat{\mu}_c)^T \hat{\Sigma}^{-1} (f(x) - \hat{\mu}_c) \quad (4)$$

To tune logistic regression parameters and  $\epsilon$ , a few ID and OOD samples are necessary. Here are the candidates for  $\epsilon$ : [0, 0.0005, 0.001, 0.0014, 0.002, 0.0024, 0.005, 0.01, 0.05, 0.1, 0.2].

**Gram** calculates the  $p$ -th order Gram matrix  $G_l^p$  with feature map  $F_l$  for every layer  $l$ . Then, correlations for any image are obtained through the Gram matrix as:

$$G_l^p = (F_l^p F_l^{pT})^{\frac{1}{p}} \quad (5)$$



Figure 2: Semantic-damaged datasets. These datasets are generated by applying OpenCV’s image processing.

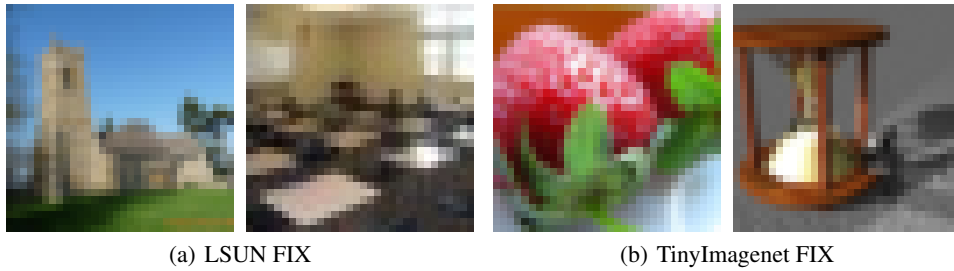


Figure 3: Semantics-preserved datasets. These datasets are generated by applying `torchvision.transforms.Resize()`.

With the Gram matrix, the class-specific minimum and maximum feature correlations are stored in an external memory. The deviation of a test image can be computed as:

$$\delta(\min, \max, value) = \begin{cases} 0 & \text{if } \min \leq value \leq \max \\ \frac{\min - value}{|\min|} & \text{if } value < \min \\ \frac{value - \max}{|\max|} & \text{if } value > \max \end{cases} \quad (6)$$

where *value* in equation 6 denotes the value appeared in the test image’s Gram matrix.

#### Transformation with Semantics-Preservation

Originally, the size of the LSUN images is uniformly  $256 \times 256$  while the images in ImageNet have various sizes. The LSUN and TinyImagenet datasets are resized to fit to the model’s input size as provided by Liang *et al.* (Liang et al., 2018). According to Tack *et al.* (Tack et al., 2020), conventional benchmarks (LSUN, TinyImagenet) contain artificial noise inserted by the OpenCV library’s image processing. It seems that the semantics in the LSUN and TinyImagenet datasets are significantly damaged. So, it is hard to recognize the picture correctly. To overcome such issue, revised datasets such as LSUN FIX, TinyImageNet FIX are generated by the PyTorch `torchvision.transforms.Resize()` operations that conduct both resizing and bilinear interpolation. Through this, the datasets preserve image semantics better than the original datasets while their resolution remains the same. This is what we

call *semantics-preserved transformation*. (See Figure 2 and 3).

#### Baseline Settings

For fair comparisons, we have selected the most commonly used options for the OOD detection task. The ResNet with 18 layers with the ReLU activation function is adopted as the model architecture. The network is trained with a batch normalization with no dropout. The model is trained for 300 epochs with a batch size of 128, the SGD optimizer with 0.9 momentum,  $5e-4$  weight decay and 0.1 learning rate which decays by a factor of 10 at  $\{0.5 \times epoch, 0.75 \times epoch\}$ . Each input is flipped randomly with a probability of 0.5, padded with edge pixels by one eighth of the size of the input, then clipped at random locations. Losses are calculated by the cross-entropy loss. Models with the best validation set accuracy were stored during the training process.