

## Appendix

In the appendix, we provide details that are omitted in the main text due to space limitation, which include:

- Details of the experiment setup.
- Details of the OoD detection algorithms.
- An additional experiment to identify important features for differentiating ID and OoD data.

### A. Details of Experiment Setup

**Software and Hardware** We implemented the deep learning models under the PyTorch framework, and performed training and testing on NVIDIA Tesla V100 GPUs. The other parts of the experiments were done on CPUs. For implementing the OC-SVM and  $k$ -NN anomaly detection algorithms, we utilized `scikit-learn` [29] and `PyOD` [42] packages; please refer to our attached code for implementation details.

**OoD Benchmarks** In our experiments, we used OoD datasets from a number of different sources as performance benchmarks. A list of these datasets and their sizes can be found in Table 5. Some of them were used by previous literature [25]. We also used public image datasets such as *Animals-10 Kaggle Dataset* [1]. In addition, we created some artificial OoD images by modifying ID images using *Jigsaw augmentation* [33]. In our Jigsaw implementation, each image was divided into  $3 \times 3$  tiles. Each tile was rotated by multiples of 90 degrees clockwise or counter-clockwise, randomly shuffled, and then recombined into a new image that is considered OoD. Furthermore, the *SVHN Dataset* [33] was used as OoD for *CIFAR10-WideResNet*, *CIFAR100-WideResNet* (Figure 7), *CIFAR10-DenseNet*, *CIFAR100-DenseNet* and *GTSRB-AlexNet* pre-trained models. We standardized both the ID and the OoD images so that their pixel values would share similar statistical properties.

**ODIN Detector** The ODIN [22] detection algorithm uses two strategies, *temperature scaling* and *input pre-processing*, for improving the OoD detection performance. Temperature scaling modifies the temperature parameter  $T$  in the softmax layer, which adjusts how strong a classifier assumes a closed world as a prior [14]. Input pre-processing adds a small controlled noise to each input to make ID and OoD sample more separable. In our experiment, we chose a noise magnitude  $\epsilon = 0.0014$  for input pre-processing.

Despite its effectiveness, ODIN requires OoD data to tune hyperparameters for both of temperature scaling and input pre-processing, which leads to two concerns: 1) the

OoD data may not always be accessible, and 2) the hyperparameters tuned with one OoD dataset may not generalize to other OoD data. To mitigate the two issues, we followed the Generalized ODIN approach proposed in follow-up work [14] and fixed the temperature parameter  $T = 1000$ .

**Mahalanobis Detector** In Lee *et al.* [20] that proposes the Mahalanobis approach, it is assumed that the features from a pre-trained network model follows class-conditional Gaussian distributions. Under this assumption, the confidence score is defined using the Mahalanobis distance with respect to the closest class-conditional distribution, where its parameters are chosen as empirical class means and tied empirical covariance of training samples. As with the ODIN approach [22], we again performed input pre-processing to better separate ID and OoD samples. We set the noise magnitude  $\epsilon = 0.0014$ .

**$k$ -NN Detector**  $k$ -NN [3, 30] is a simple, non-parametric supervised classification algorithm; it can also be used in an unsupervised fashion to detect anomalies/outliers/OoDs. The fundamental assumption of  $k$ -NN detectors is that the samples from OoD inputs tend to stay farther from the clusters of similar observations than ID ones. Given a new sample (observation) to classify, its distance to the  $k$ -th nearest neighbor in the training dataset can be viewed as the OoD score.

**OC-SVM Detector** OC-SVM [32] is another commonly used approach for anomaly or outlier detection, and thus can also be used to detect OoDs. In the OC-SVM formulation, a hyperplane is found to separate all data points in the training dataset from the origin. Therefore, given a new sample to classify, its distance from the hyperplane is treated as the OoD score.

In order to evaluate the maximum achievable performance, grid search was performed to find the best hyperparameters. We used the  $k$ -NN module from `pyOD` [42] and the OC-SVM module from `scikit-learn` [29]. from Table 7 summarizes the hyperparameters evaluated during grid-search, and Table 6 lists the hyperparameters that were selected and used in producing the results shown in Table 2.

### B. Feature Importance Analysis

We also performed a simple experiment to identify important logit entries to differentiate ID and OoD samples. Due to the limited space, we did not describe this experiment in the main paper, and deferred it to the appendix.

Assuming we have access to the OoD test data, we trained a decision tree model (a binary classifier) to differentiate between the ID and the OoD data, and analyzed

Table 5. Sizes of the OoD Datasets Used in our Experiments

Datasets	CIFAR 10 - 100 [18]	GTSRB [13] German Traffic Sign Recognition Benchmark	SVHN [26] The Street View House Numbers	ImageNet [8]
Test Set	10000	12640	26032	50000
Animals [1]	10000	10000	10000	10000
Anime Faces [6]	10000	10000	10000	-
Fishes [2]	10000	10000	10000	-
Fruits [16]	10000	10000	10000	-
iSUN [38]	8925	8925	8925	-
Jigsaw on Training Set [33]	10000	10000	10000	-
LSUN-Crop [39]	10000	10000	10000	-
LSUN-Resize [39]	10000	10000	10000	-
Office-Home Art [37]	2427	2427	2427	2427
Office-Home Clipart [37]	4365	4365	4365	4365
Office-Home Product [37]	4439	4439	4439	-
Office-Home Real [37]	4357	4357	4357	-
PACS Photo [21]	1282	1282	1282	-
PACS Art [21]	2048	2048	2048	2048
PACS Cartoon [21]	2344	2344	2344	2344
PACS Sketch [21]	3929	3929	3929	3929
Place365 [43]	10000	10000	10000	-
SVHN [26]	10000	10000	-	10000
Texture [7]	5640	5640	5640	5640

*In datasets Animals-10 Kaggle and PACS - Photo Subset, classes Cat, Dog and Horse have been removed to avoid overlap with the ID dataset*

the relative importance of each input feature. As we can see from Figure 8, the activated logits (*i.e.*, the diagonal entries in the plot) are almost always assigned the highest importance. This finding indicates that the activated logit entry is a key to differentiating the ID and the OoD data, which justifies our method to use class-wise decision rules. In addition, we also can notice that some off-diagonal logit entries also receive high feature importance scores, which motivates the use of methods that takes into account all the logit vectors (in energy-based and learning-based methods), instead of solely the maximum one, for improved detection performance.

Table 6. Selected Hyperparameters for Learning-based Detectors

	OC-SVM			$k$ -NN		
	kernel	$\nu$	$\gamma$	$k$	method	metric
WideResNet-CIFAR10	<i>poly</i>	0.1	1.0	4	<i>median</i>	<i>braycurtis</i>
DenseNet-CIFAR10	<i>poly</i>	0.4	1.0	15	<i>largest</i>	<i>braycurtis</i>
SVHN-WideResNet	<i>poly</i>	0.8	1.0	5	<i>mean</i>	<i>braycurtis</i>
GTSRB-AlexNet	<i>poly</i>	0.8	1.0	4	<i>mean</i>	<i>braycurtis</i>
WideResNet-CIFAR100 [*]	<i>poly</i>	0.1	1.0	4	<i>median</i>	<i>braycurtis</i>
DenseNet-CIFAR100 [*]	<i>poly</i>	0.4	1.0	15	<i>largest</i>	<i>braycurtis</i>
WideResNet CIFAR10 C	rbf	0.1	1.0	4	mean	<i>minkowski</i>

\* same hyperparameters of the corresponding CIFAR10 classifiers

Table 7. Hyperparameters Sweeping Intervals

$k$ -NN	#Neighbors	Metric	Method
	4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25	minkowski, braycurtis, chebyshev, euclidean, manhattan	mean, largest, median
OC-SVM	Kernel	$\nu$	$\gamma$
	poly, sigmoid, rbf, linear	0.01, 0.02, 0.03, 0.04, 0.05, 0.10, 0.15, 0.20, 0.30, 0.40, 0.50, 0.8	0.01, 0.03, 0.05, 0.08, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 1, 2, 5, 6, 7, 8, 9, 10, scale, auto

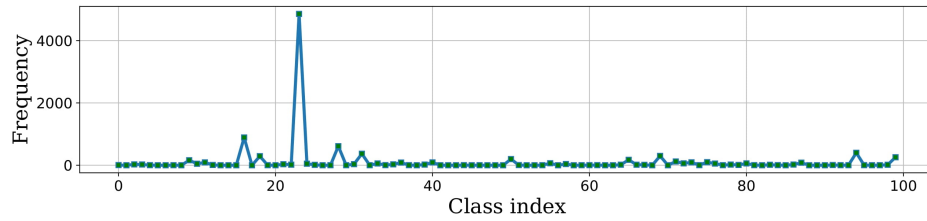


Figure 7. The uneven distribution of activated logits when the CIFAR100-pre-trained model is fed with the SVHN dataset as OoD. The most of images present in the SVHN are indeed classified as a single class, this motivate further the use of a class-wise thresholding scheme, since a given class of OoD is most likely to activate a single logit.

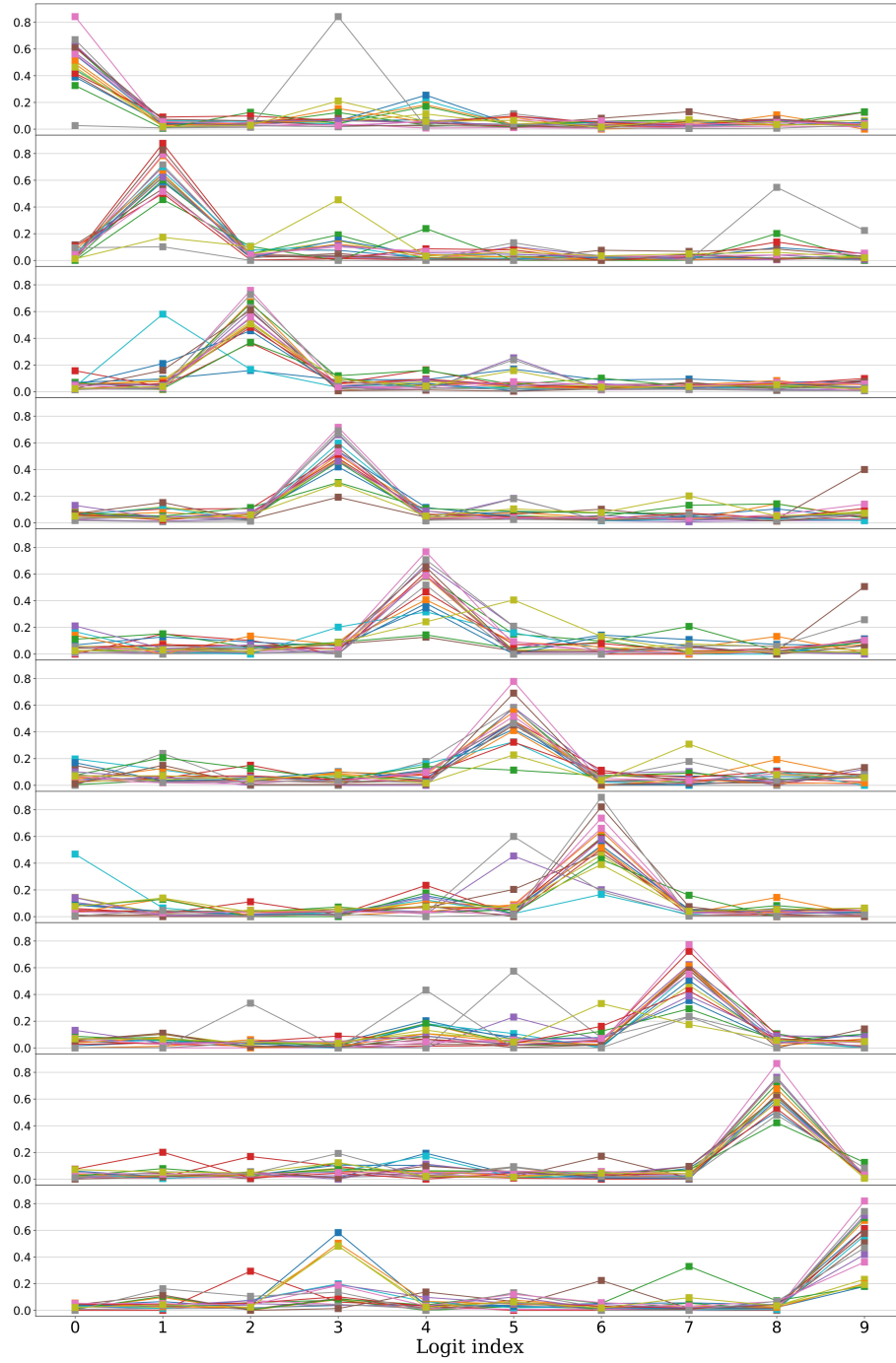


Figure 8. Gini Importance per feature, given a prediction (given argmax), CIFAR10 WideResNet

*The plot has 10 subplots, one of each represents predicted class, i.e., fixing the argmax. On each subplot on the Y-axis there is the Gini Importance, i.e. the importance that that particular feature has during the built of the tree. There are 18 lines in each subplot, for the 18 ood dataset we are considering.*