A noisy elephant in the room: Is your out-of-distribution detector robust to label noise?

Supplementary Material

In these supplementary materials we describe our experimental set-up in more detail (Section 1), along with the OOD detection methods and their hyper-parameters (Section 2). In Section 3, we present additional figures and results supporting the analysis in the main text.

1. Experimental set-up

1.1. Implementation details

Software Our implementation builds on the OpenOOD [40, 43] codebase, which provides a unified training and evaluation framework for state-of-the-art OOD detection methods. Our main changes:

- expanded the selection of classification datasets to compare the clean vs. real label noise vs. synthetic label noise settings (see Section 1.2)
- modified the selection of OOD datasets (Section 1.3)
- added the MLPMixer [35] and CompactTransformer [12] architectures (Section 1.4)
- added support for different checkpointing strategies (Section 1.4)
- modified some of the OOD detection methods' implementation to better match their official implementation (Section 2)

Compute We ran our experiments on a desktop (GTX TITAN X) and on a GPU cluster (T4, A40, V100, A100).

Reproducibility We will make our code publicly available at https://github.com/glhr/ood-labelnoise, including data splits, synthetic noisy labels, and scripts to reproduce all figures and results.

1.2. ID datasets

Overview The image classification datasets to be used for training were selected based on the availability of both clean and real noisy label sets: 3 CIFAR variants [21] and Clothing1M [39]. We also considered including datasets from the controlled real-world label noise benchmark in [19], however many of the image URLs are no longer accessible.

CIFAR10 and **CIFAR100** are widely used in OOD detection benchmarks and hardly need an introduction. They consist of natural images selected from image search engine results where the object of interest is prominent in the image and clearly identifiable - labelling was performed by students and verified by the authors [21]. **CIFAR100-Coarse** (as named in [38]) contains the same set of images as CIFAR100, but has a coarser class definition, with each superclass encompassing 5 fine-grained classes from CIFAR100. [38] provides crowd-sourced (unreliable) re-annotations of these three datasets, resulting in several noisy label sets.

Clothing1M is widely used in the label noise research bubble [1]. It consists of product images crawled from online shopping websites - with keywords in the surrounding text used to automatically assign a clothing category label. While the full noisy dataset contains over a million images, we only consider the sub-set of images which was also manually annotated by the authors, providing a corresponding clean label for every noisy label (72,409 images). Since Clothing1M includes images of variable height, they are resized to 224x224 for training and evaluation.

We show samples from these datasets in Figure 4, including the list of classes.

Train/val/test splits The validation set is used for early stopping and OOD detection hyper-parameter tuning (for methods that need it); the test set is only used for evaluation (both classification performance and OOD detection performance). Note that the only difference between the clean and label noise setting lies in the training labels - the validation and test labels are clean in both settings.

For the CIFAR datasets, we keep the same splits as in the OpenOOD benchmark [43]. For Clothing1M, we apply the official splits provided for the clean subset [39].

Noisy labels In Figure 1 we show some examples of incorrectly labeled images - the noisy labels are often quite reasonable guesses given the ambiguity of the images.

(a) Clothing1M-Noisy [39] (b) CIFAR-10-Worst-N [38] (c) CIFAR-100-Fine-N [38]

Figure 1. Examples of incorrectly labelled images from some of the real noisy training datasets.

In Figure 2 we visualize the noise transition matrix for each real noisy training set. Note that Clothing1M is the only ID dataset with class imbalance in the clean labels.

Synthetic label noise We describe the procedure for generating the 2 types of synthetic labels below. Given a set of clean and corresponding real noisy labels, the idea is to create a set of synthetic labels with **the same noise rate but following a different noise model**.

Synthetic Uniform noise (SU) labels:
given N, the number of samples to mislabel

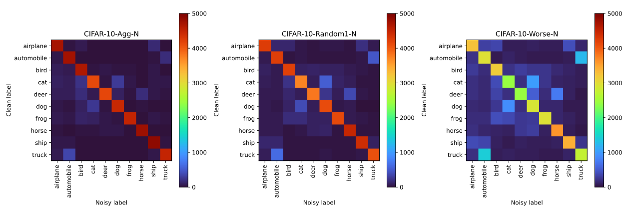
1. Select N samples from the training set randomly - these are the samples whose label we'll flip
2. For each selected sample, check its clean label, and randomly assign it one of the **other** classes with equal probability.

In Figure 3 we visualize the noise transition matrix for each SU training set.

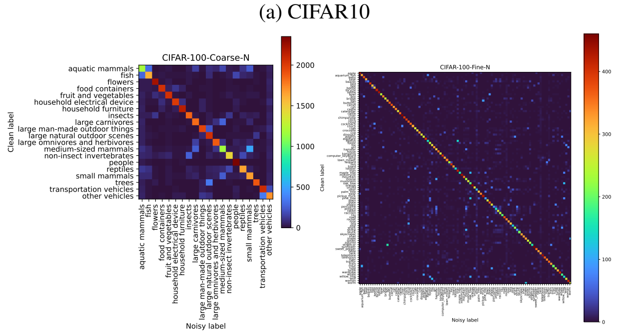
Synthetic Class-conditional noise (SCC) labels:
given the matrix indexed as M_{tf} indicating the number of samples with clean label t and noisy label f

1. For each class c, count the number of samples to mislabel: $N_{c\rightarrow} = \sum_{f \neq c} M_{cf}$
2. Select a set $S_{c\rightarrow}$ of $N_{c\rightarrow}$ samples with clean label c from the training set randomly - these are the samples whose label we'll flip
3. For each of the **other** classes (e.g. b)
   - check how many samples should be flipped to that class $N_{c\rightarrow b} = M_{cb}$
   - select $N_{c\rightarrow b}$ samples from $S_{c\rightarrow}$ randomly and flip them to class b
   - Repeat for all classes $\neq c$
4. Repeat the process for the rest of the classes

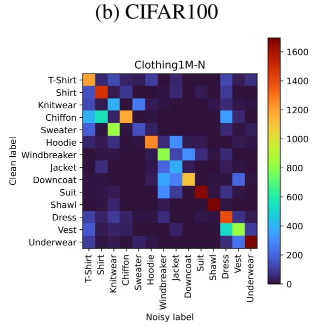
Note that the noise transition matrices for SCC datasets are identical to the corresponding real noisy label sets (Figure 2) and thus are not shown here.



(a) CIFAR10

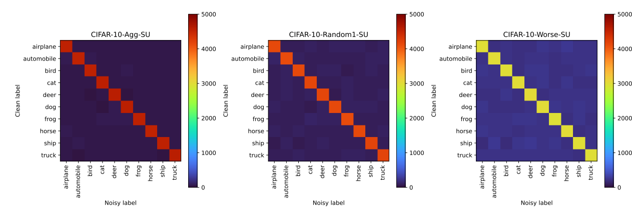


(b) CIFAR100

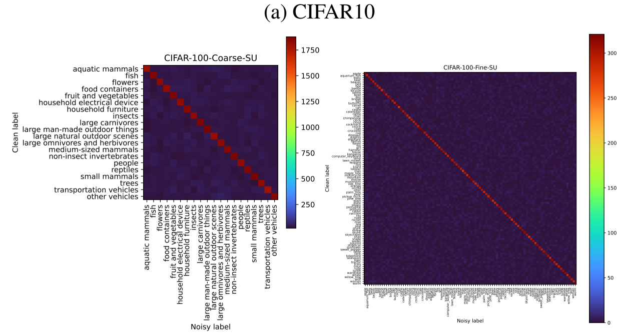


(c) Clothing1M

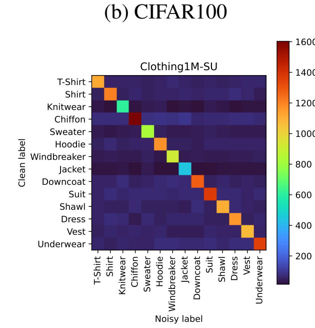
Figure 2. Confusion matrices (showing number of samples) for the **real noisy (N) training label sets**. Diagonal entries indicate the number of correctly labelled images for each class.



(a) CIFAR10

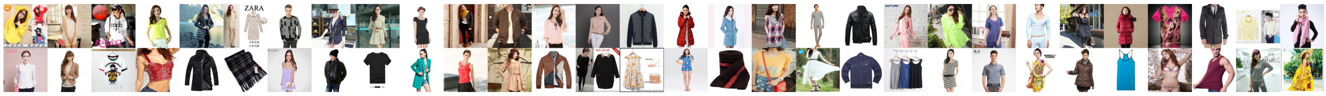


(b) CIFAR100



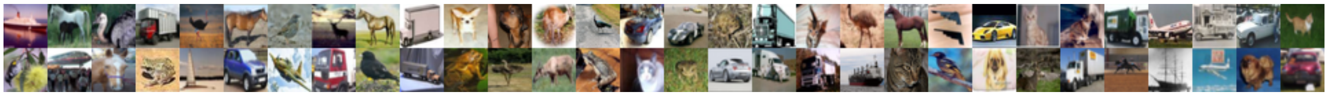
(c) Clothing1M

Figure 3. Confusion matrices (showing number of samples) for the **synthetic uniform noise** (SU) training label sets.



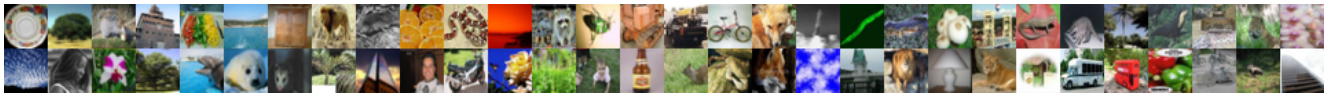
(a) **Clothing1M** [39]

14 classes: T-Shirt, Shirt, Knitwear, Chiffon, Sweater, Hoodie, Windbreaker, Jacket, Downcoat, Suit, Shawl, Dress, Vest, Underwear



(b) **CIFAR10** [21]

10 classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, trucks

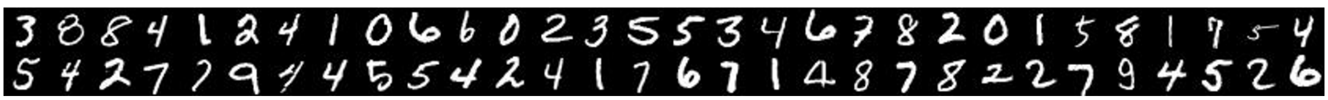


(c) **CIFAR100** [21]

20 classes for **CIFAR100-Coarse** [38]: aquatic mammals, fish, flowers, food containers, fruit and vegetables, household electrical device, household furniture, insects, large carnivores, large man-made outdoor things, large natural outdoor scenes, large omnivores and herbivores, medium-sized mammals, non-insect invertebrates, people, reptiles, small mammals, trees, transportation vehicles, other vehicles

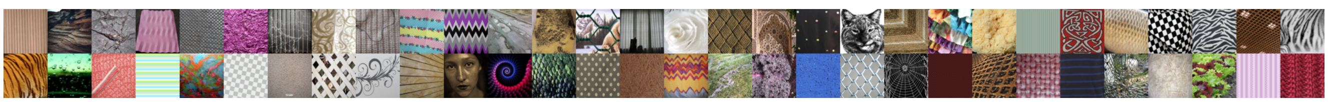
100 classes for **CIFAR100-Fine** [21]: apple, aquarium fish, baby, bear, beaver, bed, bee, beetle, bicycle, bottle, bowl, boy, bridge, bus, butterfly, camel, can, castle, caterpillar, cattle, chair, chimpanzee, clock, cloud, cockroach, couch, crab, crocodile, cup, dinosaur, dolphin, elephant, flatfish, forest, fox, girl, hamster, house, kangaroo, computer keyboard, lamp, lawn mower, leopard, lion, lizard, lobster, man, maple tree, motorcycle, mountain, mouse, mushroom, oak tree, orange, orchid, otter, palm tree, pear, pickup truck, pine tree, plain, plate, poppy, porcupine, possum, rabbit, raccoon, ray, road, rocket, rose, sea, seal, shark, shrew, skunk, skyscraper, snail, snake, spider, squirrel, streetcar, sunflower, sweet pepper, table, tank, telephone, television, tiger, tractor, train, trout, tulip, turtle, wardrobe, whale, willow tree, wolf, woman, worm

Figure 4. Preview of the ID datasets - with 60 randomly selected training samples shown per dataset.

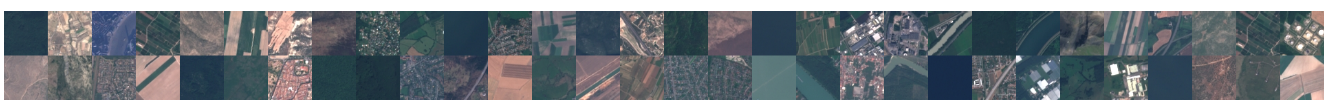


(a) MNIST [8] (**OOD test**)

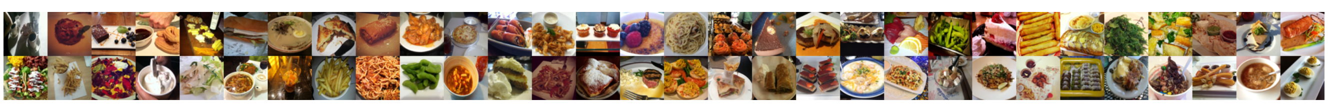
(b) SVHN [27] (**OOD test**)



(c) Textures [5] (**OOD test**)

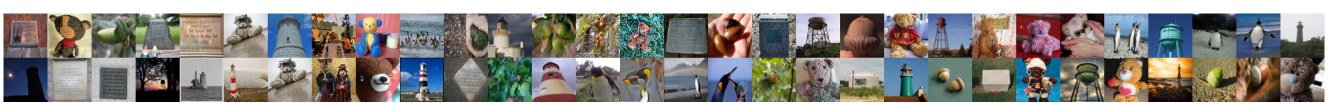


(d) EuroSAT [15] (**OOD test**)

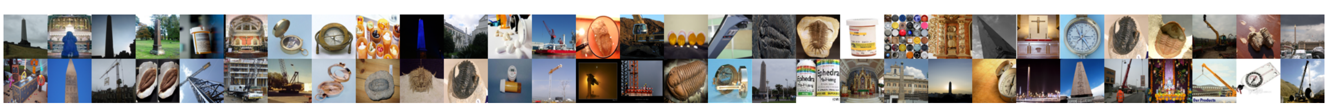


(e) Food-101 [4] (**OOD test**)

(f) Stanford Online Products [30] (**OOD test**)



(g) ImageNet sub-set (**OOD test**)



(h) ImageNet sub-set (**OOD val**)

Figure 5. Preview of the OOD test and validation sets - with 60 randomly selected samples shown per dataset.

1.3. OOD datasets

Figure 5 shows samples from each OOD dataset. Our selection of OOD datasets differs from that of the CIFAR-10 and CIFAR-100 benchmarks from OpenOOD [43], since we wanted to minimize the possibility of semantic overlap with any of the ID datasets. We did not include Places365 [44] as it prominently features people, overlapping with Clothing1M and CIFAR100. Instead, we include **Food101** [4], **EuroSAT** [15], and selected classes[1] from **Stanford Online Products** [30] - these three datasets provide additional semantic and appearance diversity from different domains. We kept **MNIST** [8], **SVHN** [27], **Texture** [5] as they significantly differ from the ID datasets both semantically and appearance-wise. We also include **12 classes from TinyImagenet** [22]**/ImageNet** [7] - which we manually selected to be person-free and not overlap with the ID datasets. 6 of those classes[2] are randomly selected as OOD *validation set* (only used for hyper-parameter tuning) - and the rest[3] are used as an OOD test set - alongside the 6 other OOD datasets.

1.4. Training classifiers

Architectures and hyper-parameters We selected 3 classifier architectures with two main objectives in mind: including several families of neural networks, and keeping the experiments feasible in terms of time and compute.

- The **ResNet18** [14] model implementation and training hyper-parameters are fully based on the OpenOOD benchmark, with the `32x32` variant used for CIFAR datasets and `224x224` for Clothing1M. We used the same training hyper-parameters for all datasets.
- The **Compact Transformer** [12] model implementation and training hyper-parameters are based on the official repository [13]. We use the `cct_7_3x1_32` variant for CIFAR datasets, and `cct_7_7x2_224` for Clothing1M. We train the models with AdamW optimization and cosine annealing - learning rates 5.5e-4 for CIFAR10, 6e-4 for CIFAR100, and 5e-4 for Clothing1M (based on the hyper-parameters in the official repository).
- The **MLPMixer** [35] model implementation and training hyper-parameters are based on a third-party repository[41]. For the CIFAR datasets, a patch size of 4, network depth of 6 and feature dimensionality of 512 are used; for Clothing1M, a patch size of 16, and depth of 8. Models are trained via Adam optimization with learning rate 1e-3 and cosine annealing.

[1] stapler, toaster, coffee maker, cabinet, fan, kettle

[2] magnetic compass, lighthouse, water tower, trilobite, obelisk, penguin, crane, altar, brass, acorn, teddy bear, pill bottle

[3] crane, pill bottle, magnetic compass, obelisk, altar, trilobite

Following the OpenOOD training pipeline, besides normalization and standard data augmentation, no advanced training techniques were applied.

Label noise setting Models trained on noisy labels (real or synthetic) are trained with the same images, hyper-parameters and procedure as their cleanly trained counterpart. Thus, the only difference lies in the training labels used to compute the loss for training.

Early stopping We monitor validation accuracy (using clean validation labels) every epoch, and save two checkpoints during training:

1. *early* - epoch which gave the top-1 validation accuracy.
2. *last* - last epoch (we set a pre-defined number of epochs for each architecture which is based on the number of ecochs in their respective implementations and ensures convergence - 100 epochs for ResNet18, 300 epochs for Compact Transformer and 500 epochs for MLPMixer).

1.5. Evaluation

Classification performance At test-time, a model is first evaluated on its ID test set D_{test} (e.g. CIFAR10 test for a classifier trained on CIFAR10 images) in terms of classification accuracy. We denote the set of ID test images which are correctly classified as $D_{correct}$, and the others as $D_{incorrect}$.

OOD detection performance Next, each of the 20 OOD detection methods is applied to the classifier (with some methods requiring a set-up step to collect ID data statistics, and others requiring a hyperparameter tuning step using validation ID and OOD samples - see details in Section 2). OOD detector scores are then extracted for all samples in D_{test}, as well as all samples in each of the 7 OOD test datasets. Note that no samples from D_{test} or from the OOD test sets are "seen" by the classifier or OOD detector before evaluation. For each OOD test dataset (e.g. D_{MNIST}), we measure:

- $\text{AUROC}_{\text{ID vs. OOD}}$ - where D_{test} samples are considered positive and D_{MNIST} are considered negative
- $\text{AUROC}_{\text{correct vs. OOD}}$ - where $D_{correct}$ samples are positive and D_{MNIST} samples are negative ($D_{incorrect}$ samples are excluded from the calculation)
- $\text{AUROC}_{\text{incorrect vs. OOD}}$ - where $D_{incorrect}$ samples are positive and D_{MNIST} samples are negative ($D_{correct}$ samples are excluded from the calculation)

OOD detection performance is aggregated across the 7 OOD test datasets when reporting results - we report the median AUROC in the main text as it is less sensitive to outliers than the mean. For completeness, we also include results for mean AUROC (Section 3.1 below).

Lastly, in this supplementary we also report the AUROC$_{\text{correct vs. incorrect}}$ - that is, the OOD detectors' ability to flag misclassifications among ID samples (known as failure detection in the literature [3]). Note that AUROC is not sensitive to the number/ratio of positive vs. negative samples, as the ROC curve plots the True Positive rate vs. False Positive rate.

1.6. Statistical testing

When comparing pairs of methods (e.g. MSP vs. ODIN) or settings (clean vs. noisy training labels), we apply the Almost Stochastic Order (ASO) test [6, 10] as implemented by Ulmer et al. [36] with the `deepsig` Python package: https://github.com/Kaleidophon/deep-significance.

ASO is a statistical significance test which enables pairwise comparison of two sets of scores - one produced by model/method/setting A, and the other by model/method-/setting B (the baseline). It compares their empirical distributions and determines whether A can be declared *stochastically dominant* over B by comparing the empirical cumulative distribution functions of the two sets of scores. It is well suited for deep learning research, since it does not make any assumptions about the distribution of a set of scores.

The ASO test is parametrized by a threshold α - the significance level that the p-value has to fall below (we use $\alpha = 0.05$). It outputs a value $\epsilon_{\min}$ (ranging from 0 to 1) corresponding to the (expected upper bound to the) violation ratio. If $\epsilon_{\min}[\text{A}>\text{B}] < 0.5$, then A can be considered stochastically dominant over B (A's scores are bigger than B's more often than not). The lower $\epsilon_{\min}[\text{A}>\text{B}]$, the more confident we can be that A is superior. If $\epsilon_{\min}[\text{A}>\text{B}] \geq 0.5$, we cannot consider A superior to B. In that case, only by testing $\epsilon_{\min}[\text{B}>\text{A}]$ can we say whether B is superior to A. If $\epsilon_{\min}[\text{A}>\text{B}] \geq 0.5$ *and* $\epsilon_{\min}[\text{B}>\text{A}] \geq 0.5$, neither A nor B can be considered superior to the other. We refer to [36] for additional details about ASO and its implementation.

In our case, the scores correspond to AUROC performance for different runs (e.g. across multiple random seeds and checkpointing strategies).

2. OOD detection methods

How were the methods chosen? Our selection of OOD detection methods includes all post-hoc methods from the OpenOOD v1.5 benchmark [43], excluding OpenGAN [20] as it requires training a secondary network (too computationally demanding considering the scale of our experiments), but including GEN [26] which was recently added to the OpenOOD codebase but is not mentioned in the OpenOOD papers.

Configuration and hyper-parameters Table 1 gives an overview of the 20 methods. Many methods first involve a set-up step where one or more parameters are adjusted based on training or validation ID data. Some methods are also governed by hyper-parameters, which are tuned based on OOD detection performance on validation data.

In most cases, we follow the OpenOOD implementation/settings for each method, and refer to the OpenOOD paper and codebase for details. For some methods, we modified their implementation or configuration, or expanded their set of hyper-parameters (bold in Table 1). We describe these modifications below.

- ODIN [24] - we extended the range of possible values for the perturbation magnitude to align with the original paper and implementation.
- ASH [9] - the original paper proposes two competitive variants: ASH-s and ASH-b, and OpenOOD implements ASH-b by default. We included both in our preliminary experiments and found ASH-s to perform significantly better than ASH-b (both in a clean and noisy label setting), and thus only report results for ASH-s.
- REACT [33] and DICE [32] - we extended the range of possible values for the percentile parameter based on results in the original papers.
- MDSEnsemble [23] - the OpenOOD implementation only extracts features from the first layer - we instead extract features after every layer, following the original paper. We did not apply any input perturbation because we wanted to isolate the effect of ensembling (for comparison with MDS), and because it would require a hyper-parameter tuning step to tune the perturbation magnitude (time- and compute-hungry).
- GRAM [29], the OpenOOD implementation gave subpar results - we therefore re-implemented it following the official implementation, and extract features after every layer similarly to MDSEnsemble. Furthermore, when computing class-wise statistics in the setup it is assumed that there is at least one prediction per class - however, in the label noise setting we found a few corner cases where a class has no prediction in the training set. In those cases, we select samples for that class based on class labels.
- OpenMax [2] - we added a fallback for when a class has no corresponding predictions, similarly to GRAM.
- SHE [42] - in the setup it is assumed that there is at least one correct prediction per class (to obtain a mean activation per class), which again, sometimes is not the case for a class. When this corner case happens, we instead consider samples which are either predicted or labelled as that class.

methods	*set-up using ID data?*	*hyper-parameters* (tuned to maximize AUROC btw. ID val set and OOD val set)	*configuration details*
ODIN [24]		temperature in $\{1, 10, 100, 1000\}$ **perturbation magnitude in** $\{\mathbf{0}, \mathbf{0.00035}, \mathbf{0.0007}, \mathbf{0.0014}, \mathbf{0.0028}\}$	
GEN [26]		gamma in $\{0.01, 0.1, 0.5, 1, 2, 5, 10\}$ M in $\{10, 50, 100, 200, 500, 1000\}$	
ASH [9]		percentile in $\{65, 70, 75, 80, 85, 90, 95\}$	**ASH-s variant**
EBO [25]		temperature in $\{1, 10, 100, 1000\}$	
REACT [33]	✓	**percentile in** $\{\mathbf{80}, \mathbf{85}, \mathbf{90}, \mathbf{95}, \mathbf{99}\}$	
DICE [32]	✓	**percentile in** $\{\mathbf{10}, \mathbf{30}, \mathbf{50}, \mathbf{70}, \mathbf{90}\}$	
KNN [34]	✓	K in $\{50, 100, 200, 500, 1000\}$	
MDSEnsemble [23]	✓		**every layer, no input perturbation, weight of 1 for all layers**
GRAM [29]	✓		**every layer, powers in range [1,10], weight of 1 for all layers**
OpenMax [2]	✓		Euclidean distance, Weibull tail size of 20, alpha rank of 3
SHE [42]	✓		inner product as distance function
RMDS [28]	✓		
KLM [17]	✓		
MDS [23]	✓		
VIM [37]	✓		
TempScaling [11]	✓		
GradNorm [18]			
RankFeat [31]			
MLS [17]			
MSP [16]			

Table 1. Implementation and configuration details for each OOD detection method. Entries in bold indicate that we modified it compared to the OpenOOD implementation.

3. Analysis

Here we delve into the results, following a similar outline and sub-section titles as in the main text, for easy cross-referencing.

3.1. Where there's noise there's trouble

Performance on each OOD dataset In the main text, we aggregate OOD performance across the 7 OOD test datasets; Figure 6 shows the effect of noisy vs. clean classifier labels for each individual OOD dataset. Across the board, ImageNet6$_{\text{test}}$ and Stanford Online Products are the most challenging OOD datasets, while EuroSAT, SVHN and MNIST tend to be easier.

Methods with the strongest potential in a label noise setting (GRAM, KNN, MDS, MDSEnsemble and VIM) are nevertheless sensitive to the characteristics of the OOD dataset - the spread in performance across OOD datasets is especially large for GRAM and MDSEnsemble which operate at multiple network depths.

We also note that the drop in OOD detection performance caused by the introduction of label noise tends to be more pronounced for OOD datasets with the highest *clean* performance - in other words, label noise reduces the performance gap between OOD datasets of varying difficulty.

Aggregating performance across OOD datasets For almost all methods, aggregating OOD detection performance by taking the mean vs. median AUROC across OOD datasets gives comparable results ($\epsilon_{\min}$[mean>median] $>$ 0.5 and $\epsilon_{\min}$[median>mean] $>$ 0.5) - the exception being MDSEnsemble, for which the median AUROC is consistently larger than the mean AUROC ($\epsilon_{\min}$[median>mean] $=$ 0.15). This is also reflected in the best-case performance reported in Table 2, where MDSEnsemble ranks significantly lower than in Table 2 from the main text, and in Figure 6 where its strong performance is limited to far OOD datasets. However, in terms of best-case performance (Table 2) it remains the strongest method for classifiers trained on Clothing1M images.

In the following sections, we continue to report the median AUROC in the rest of the supplementary, aligning with the main text.

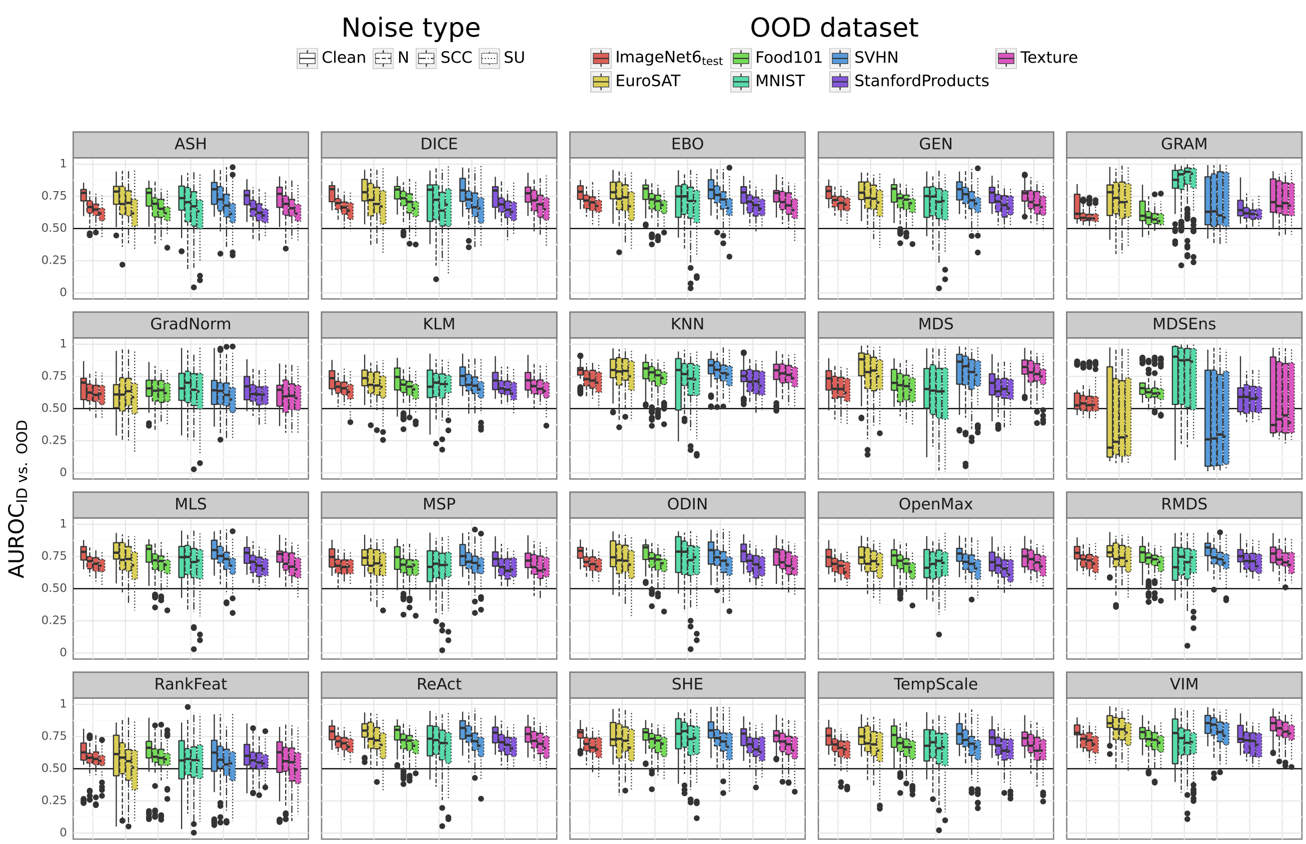


Figure 6. OOD detection performance across different OOD datasets and training label types. Clean: reliable labels, N: real noisy labels, SCC and SU: synthetic noisy labels. Boxplots show the distribution across models/checkpoints.

training labels	CIFAR10										CIFAR100-Coarse				CIFAR100-Fine				Clothing1M			
		Agg			Rand1			Worst														
method	**clean**	**N**	**SCC**	**SU**	**N**	**SCC**	**SU**	**N**	**SCC**	**SU**	**clean**	**N**	**SCC**	**SU**	**clean**	**N**	**SCC**	**SU**	**clean**	**N**	**SCC**	**SU**
MDS	94.08	**88.7**	**88.2**	89.85	89.14	88.87	87.43	87.07	87.17	87.44	82.28	76.19	79.01	78.06	76.28	74.04	73.51	**71.86**	84.99	**87.61**	87.9	**88.99**
VIM	**93.84**	**88.85**	87.87	**89.48**	**88.69**	**88.66**	**86.42**	**86.52**	**86.62**	**86.93**	83.32	**75.74**	**78.23**	**76.76**	81.49	**73.21**	73.83	71.59	**86.82**	82.84	**88.01**	84.98
KNN	**93.29**	89.11	89.13	**88.92**	**87.26**	**86.78**	86.22	85.11	**84.88**	**85.03**	***82.81***	***74.16***	**77.41**	73.96	**82.78**	**76.55**	73.84	69.19	84.45	**84.78**	**87.91**	83.37
GRAM	91.19	84.97	83.16	83.12	***81.86***	82.12	82.31	80.27	81.4	80.41	80.83	74.17	74.44	73.6	83.8	77.77	76.68	75.93	**87.63**	82.69	86.45	**86.32**
RMDS	**92.28**	**88.19**	**88.85**	**88.22**	87.88	**85.64**	**86.85**	**84.52**	**81.5**	**82.23**	81.43	**75.51**	**76.39**	**74.14**	82.06	**76**	**76.67**	**72.56**	**73.79**	**70.84**	**79.18**	**71.82**
DICE	90.03	83.15	81.77	84.34	***87.74***	80.91	84.61	83.45	76.58	78.45	**82.48**	73.56	74.74	70.41	80.83	73.57	72.95	70.74	85.64	79.64	84.09	86.05
EBO	***90.94***	***85.73***	***84.87***	***82.22***	**88**	***82.41***	**77.65**	**85.88**	**82.38**	***80.92***	82.08	**74.95**	72.95	**68.51**	80.47	***74.19***	**72.61**	**67.09**	**86.41**	***79.64***	**86.05**	**71.52**
GEN	91.05	**85.43**	84.55	82.62	87.84	82.83	82.02	85.84	81.23	81.59	**81.7**	75.04	**72.06**	70.09	80.35	73.53	**73.98**	67.74	82.48	75.32	82.81	70.63
ODIN	**90.97**	86.7	**85.59**	**82.87**	**87.52**	**82.61**	**82**	**85.21**	**81.95**	**81.17**	80.43	**74.43**	**71.16**	**69.41**	82.62	74.23	72.5	67.4	82.89	**75.51**	**80.07**	**70.44**
ReAct	90.29	**86.02**	85.17	82.49	87.63	82.99	80.61	85.66	***79***	79.61	***82.25***	**75.14**	***73.11***	***71.04***	***82.87***	**75.18**	***74.17***	**66.15**	**82.38**	**73.44**	**80.81**	**71.66**
SHE	**89.79**	85.93	**84.68**	**85.15**	**86.46**	**83.49**	***83.08***	***83.41***	***81.16***	**80.87**	80.45	***71.44***	**75.23**	69.06	77.69	69.53	68.7	67.84	***85.65***	**79.79**	***81.31***	***75.38***
MLS	90.85	85.64	84.75	82.16	87.17	82.9	78.85	85.21	82.05	81.07	81.92	74.85	71.83	69.07	80.41	74.03	72.5	67.63	81.94	74.54	80.08	70.2
MDSEns	92.42	84.48	80.9	80.53	79.21	80.2	79.04	77.21	79.04	79.7	71.52	66.99	65.56	65.89	**77.94**	71.01	70.33	69.73	91.34	91.26	91.52	91.57
TempScale	90.83	85.13	84.22	82.34	84.54	79.19	79.97	83.73	80.44	81.07	80.59	72.97	70.3	69.14	**79.6**	73	69.21	67.25	78.49	69.75	86.15	70.71
ASH	88.02	84.31	82.86	80.5	82.14	74.98	**74.81**	80.62	76.43	76.38	81.98	72.86	73.42	65.69	82.12	75.31	72.3	67.61	82.22	77.4	79.53	75.22
MSP	90.62	84.76	84.12	82.45	84.59	81.99	***82***	83.13	79.72	81.03	79.93	71.49	68.95	69.01	***78.73***	71.11	***68.65***	67.49	76.42	66.75	74.26	70.72
OpenMax	89.79	85.77	83.02	82.87	82.88	82.04	78.65	79.95	75.42	78.41	**80.88**	74.75	72.5	69.69	**80.1**	***74.67***	**74.19**	68.89	70.28	67.94	75.15	69.53
KLM	90.63	82.52	82.41	82.67	80.23	80.89	82.26	76.9	77.47	77.32	***78.99***	71.22	69.52	67.97	79.13	72.1	70.75	66.69	75.75	66.65	65.45	66.54
GradNorm	85.5	78.96	77.03	76.46	81.92	79.79	78.08	79.1	75.1	74.81	68.4	67.05	69.42	66.22	71.05	64.9	66.97	62.77	82.81	77.01	77.99	72.54
RankFeat	**81.41**	**77.82**	**78.65**	**72.14**	77.1	**74.92**	75.72	**81.36**	**75.86**	**74.64**	**74.88**	64.79	68.15	**64.76**	68.32	70.27	66.97	**65.93**	**71.63**	73.57	72.35	69.04

Table 2. **Best-case** OOD detection performance ($\text{AUROC}_{\text{ID vs. OOD}}$ in % when taking the **mean** AUROC across OOD datasets, as opposed to the median in the main text) per method (that is, after selecting the best architecture-seed-checkpoint combination for each training label set). N, SCC, and SU refer to the real and synthetic noisy label sets described in in the main text. The top-3 for each training dataset are highlighted in bold, and the top-1 is underlined. In red are scores $< 75\%$ and in orange scores between 75 and 80%. Rows are sorted based on the total performance across columns.

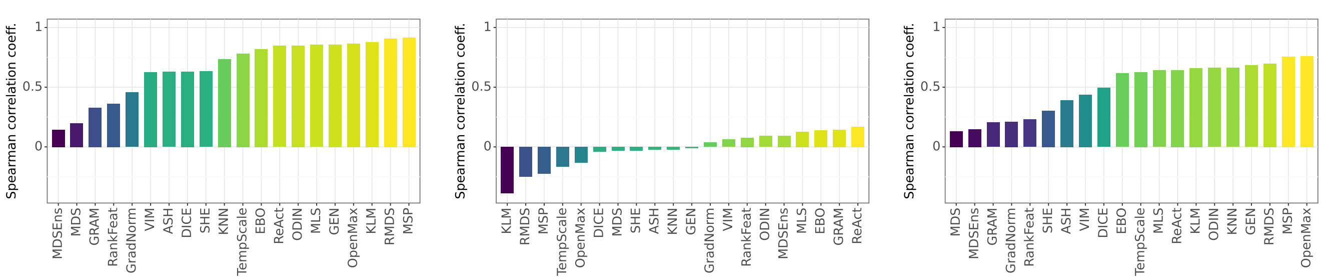
3.2. Does accuracy tell the whole story?

Relation between OOD detection performance and accuracy Figure 7 summarizes the relation between OOD performance metrics and classification accuracy in terms of Spearman correlation. As mentioned in the main text, the correlation does not hold when only considering incorrectly classified ID samples, which the majority of methods are not capable of distinguishing from OOD samples regardless of the underlying classifier. The gap between $\text{AUROC}_{\text{correct vs. OOD}}$ and $\text{AUROC}_{\text{incorrect vs. OOD}}$ for each method is shown in Figure 10 (next section).

In Figure 8 we visualize the OOD detection performance of individual models as a function of ID classification accuracy, separately considering $\text{AUROC}_{\text{correct vs. OOD}}$ performance (left) vs. $\text{AUROC}_{\text{incorrect vs. OOD}}$ (right), and color-coding points according to different parameters of interest.

For instance, Figure 8c shows the role of the classifier's architecture. Methods taking logits or Softmax probabilities as input are not visibly affected by the choice of architecture. GRAM, MDS and MDSEnsemble's performance is clearly architecture-dependent. Notably, MDSEnsemble and RankFeat's $\text{AUROC}_{\text{correct vs. OOD}}$ performance drops below 0.5 (worse than a random detector) when using a CompactTransformer architecture (CCT). Looking at Figure 8b, we note that TempScale applied on a classifier trained with synthetic noisy labels can give unexpectedly poor results, even at low noise rates (Figure 8a).

Looking at the *spread* of $\text{AUROC}_{\text{correct vs. OOD}}$ performance in Figure 8c, it appears that even for methods with the highest correlation, OOD detection performance becomes less predictable as classification accuracy decreases.

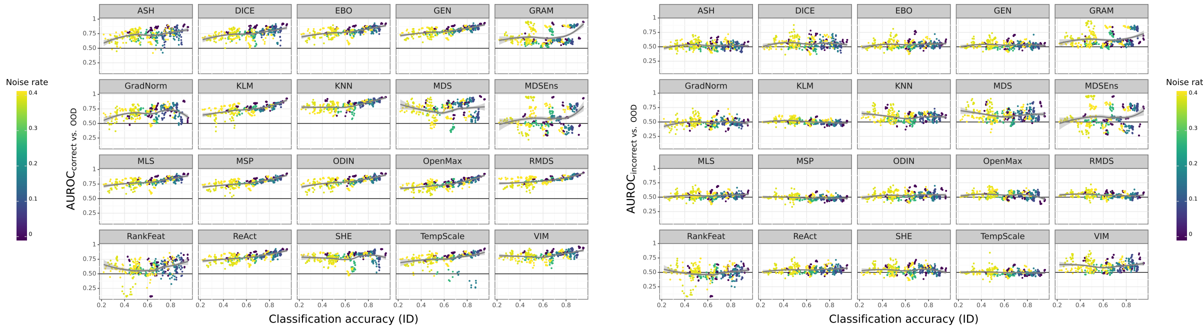


(a) Correlation between ID classification accuracy and $\text{AUROC}_{\text{ID vs. OOD}}$

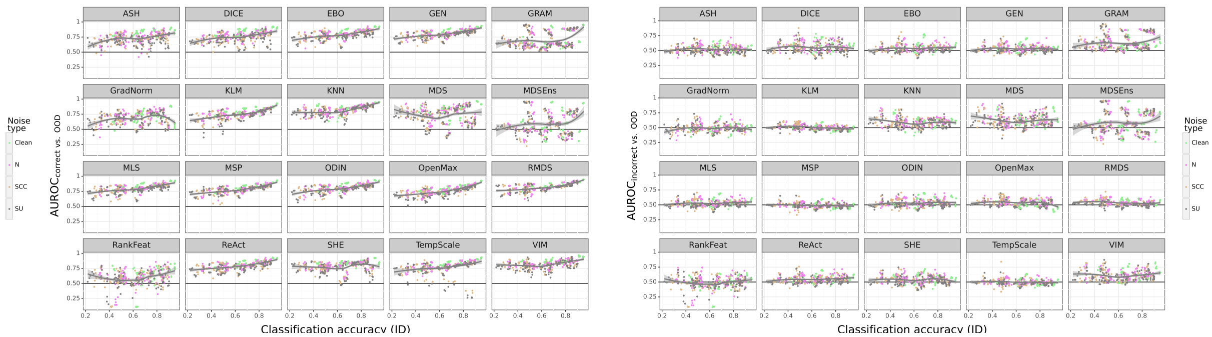
(b) Correlation between ID classification accuracy and $\text{AUROC}_{\text{incorrect vs. OOD}}$

(c) Correlation between ID classification accuracy and $\text{AUROC}_{\text{correct vs. OOD}}$

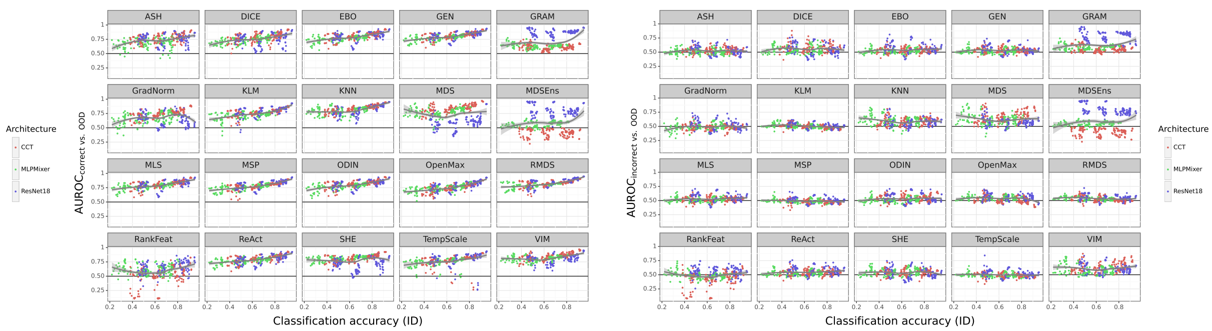
Figure 7. Spearman correlation between ID classification and OOD detection performance across methods. Each model/checkpoint contributes a single point (396 points per method).



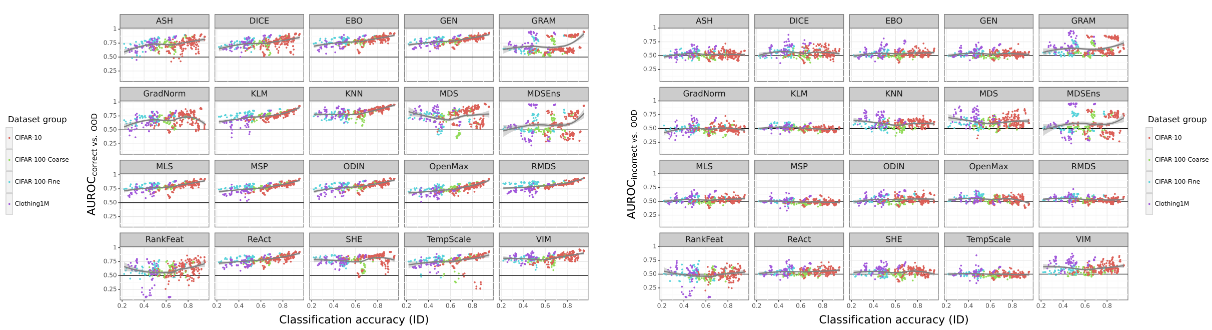
(a) Color-code by noise rate (0 corresponds to clean training labels)



(b) Color-code by noise type



(c) Color-code by architecture



(d) Color-code by training dataset group

Figure 8. Exploring the relationship between OOD detection performance and ID classification performance (accuracy). Here we distinguish between the OOD detectors' ability to separate *correctly classified* ID samples with OOD samples (left) and *incorrectly classified* ID samples with OOD samples (right). Each point represents a single model/checkpoint (396 points per method).

Would your OOD detectors be better off as a failure detector? In the main text we raise this question in passing, and do not elaborate on failure detection performance due to lack of space. Here, we delve deeper and show supporting results.

Performing pair-wise statistical comparisons between failure detection performance ($\text{AUROC}_{\text{correct vs. incorrect}}$) and OOD detection performance ($\text{AUROC}_{\text{ID vs. OOD}}$) across all models/checkpoints for each method (Figure 9), we find:

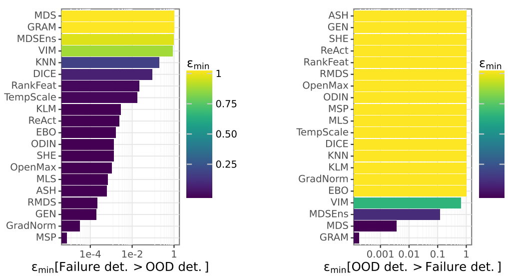
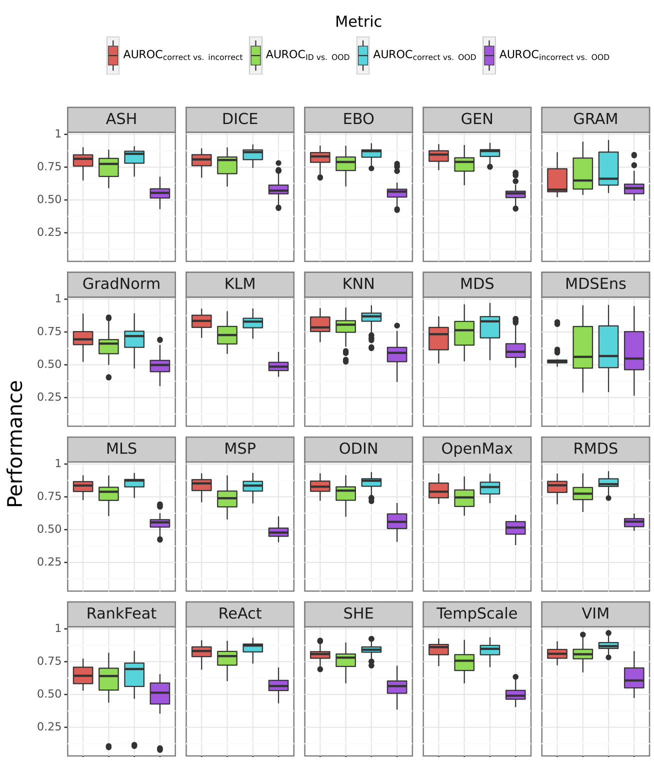


Figure 9. Almost Stochastic Order (ASO) [6, 10, 36] pairwise comparisons between failure detection and OOD detection performance for each method. The smaller $\epsilon_{\min}$[A>B], the more confident we can be that setting A is better than setting B.

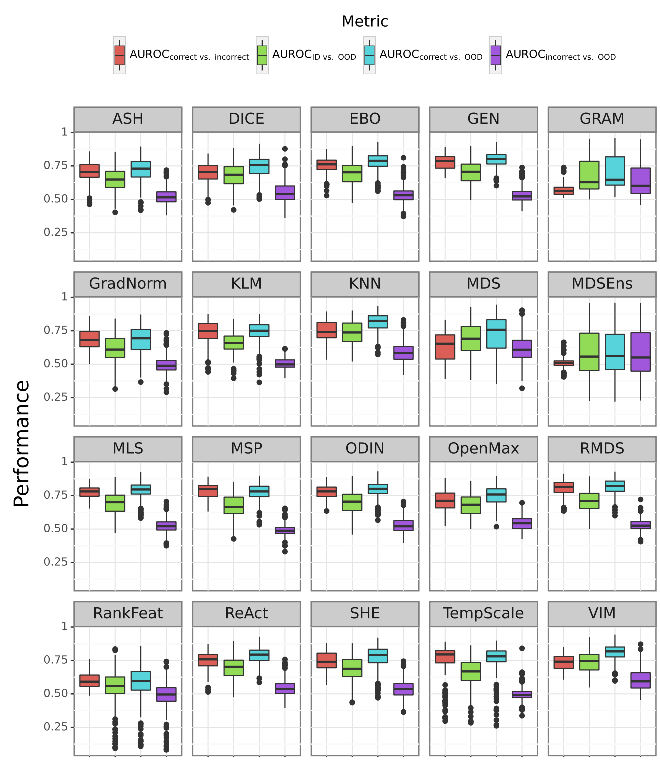
- MSP, GradNorm, GEN, RMDS, ASH, MLS, OpenMax, SHE, ODIN, EBO, REACT, KLM are all consistently better failure detectors than OOD detectors ($\epsilon_{\min}$[failure detection>OOD detection] $< 10^{-2}$)
- the same holds for TempScaling, Rankfeat, DICE and KNN, although less consistently ($\epsilon_{\min}$[failure detection>OOD detection] < 0.2)
- VIM is the only method which cannot be considered better at one or the other ($\epsilon_{\min}$[failure detection>OOD detection] $= 0.86$ and $\epsilon_{\min}$[OOD detection>failure detection] $= 0.65$)
- GRAM, MDS, and MDSEnsemble are the only methods which are better OOD detectors than failure detectors ($\epsilon_{\min}$[OOD detection>failure detection] < 0.13)

Figure 10 provides a visual comparison of these 2 metrics (red and green boxplots) in the clean and label noise setting. For most methods, the gap between failure detection and OOD detection performance is clear even in a clean label setting, and widens with the introduction of label noise in the classifier's training data.

TLDR; indeed, many state-of-the-art post-hoc OOD detectors would be better off as failure detectors.



(a) Clean label setting



(b) Noisy label setting

Figure 10. Comparing the performance of OOD detectors across different AUROC-based evaluation metrics in the clean vs. label noise setting. $\text{AUROC}_{\text{correct vs. incorrect}}$ (in red) is a *failure detection* metric - it only considers ID samples. The rest are OOD detection metrics. Boxplots show the distribution across all models/checkpoints.

It's not just about the noise rate Figure 6 (presented earlier) breaks down the effect of noise type per method and per OOD dataset, allowing for a fine-grained visual comparison. Figure 11 gives a more concise overview, showing a clear trend across methods that synthetic labels are more detrimental than real noisy labels despite having the same noise rate. GRAM and MDSEnsemble, the two ensemble-based methods are the least sensitive to the noise model.

Figure 12 looks at the effect of label noise in terms of how ID and OOD scores are distributed (similarly Figure 5 in the main text). Note that the images used to compute scores are the same across histograms, only the underlying classifier's training labels differ. Separability between ID and OOD samples clearly degrades with increasing label noise. These examples also show that for a given noise rate, the distribution of label noise across samples and classes affects the magnitude (x-axis) and spread of ID and OOD scores.

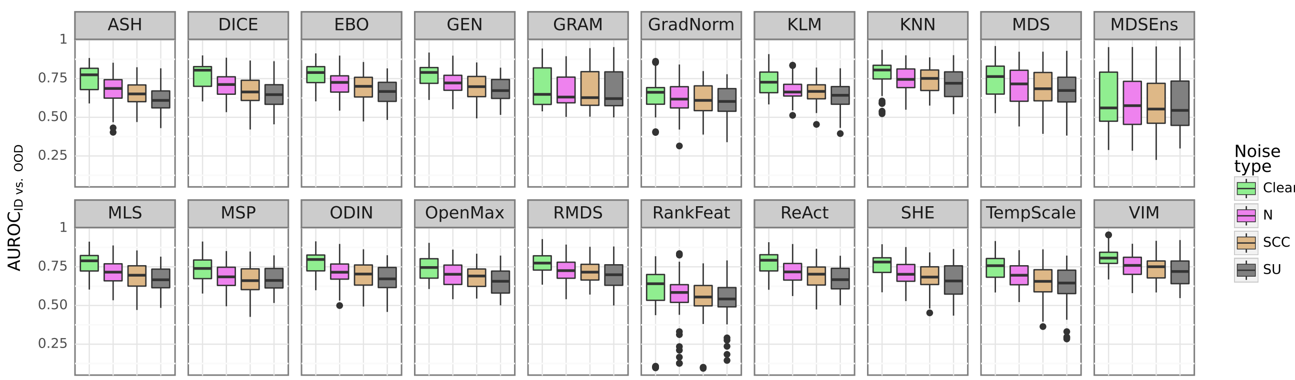
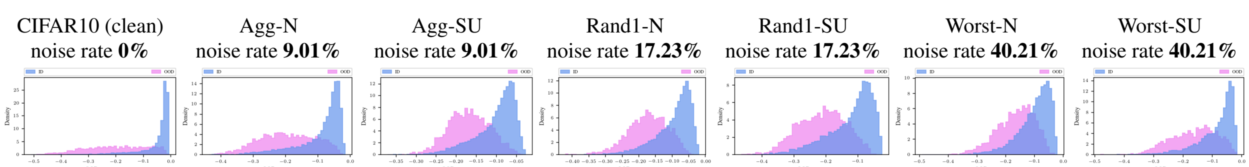
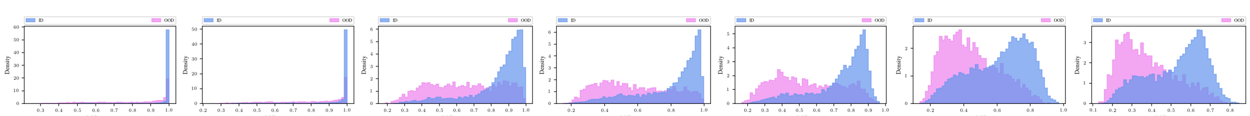


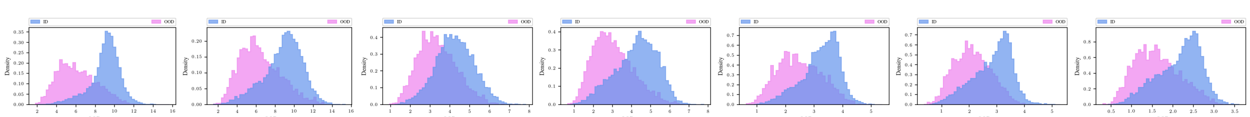
Figure 11. OOD detection performance for different types of labels used to train the underlying classifier. Clean: reliable labels, N: real noisy labels, SCC and SU: synthetic noisy labels.



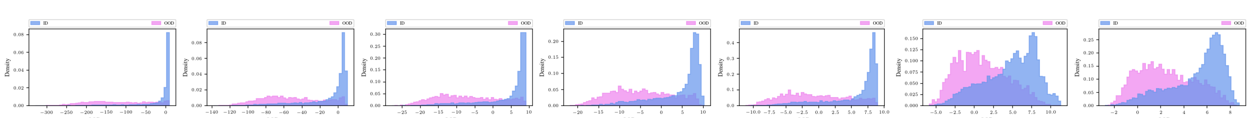
(a) KNN



(b) MSP



(c) MLS



(d) RMDS

Figure 12. Histogram of OOD detector scores for ID samples from the CIFAR10 test set (blue) and OOD samples from ImageNet6$_{\text{test}}$. We consider different OOD detectors placed on top of a ResNet18 classifier (early checkpoint) trained with different sets of labels (indicated on top of the histograms). Note that x axis limits are adjusted for each histograms.

Picking a model checkpoint Figure 13 compares performance when using clean validation set or the noisy training set for set-up/tuning.

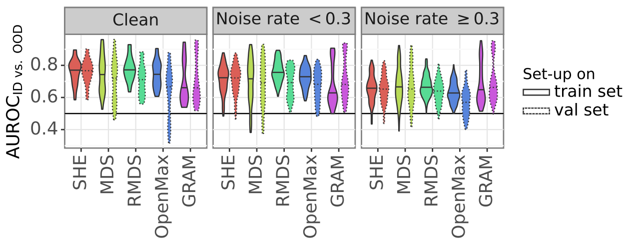


Figure 13. OOD detection performance when using clean validation labels (dashed line) vs. potentially noisy labels from the training set (solid line) for set-up. In the "clean" setting, both sets of labels are reliable, and differ in terms of size/amount but not quality.

References

[1] Görkem Algan and Ilkay Ulusoy. Image classification with deep learning in the presence of noisy labels: A survey. *Knowledge-Based Systems*, 215:106771, 2021. 1

[2] Abhijit Bendale and Terrance E. Boult. Towards open set deep networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1563–1572, 2016. 5, 6

[3] Mélanie Bernhardt, Fabio De Sousa Ribeiro, and Ben Glocker. Failure detection in medical image classification: A reality check and benchmarking testbed. *Transactions on Machine Learning Research*, 2022. 5

[4] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014. 3, 4

[5] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, 2014. 3, 4

[6] Eustasio Del Barrio, Juan A Cuesta-Albertos, and Carlos Matrán. An optimal transportation approach for assessing almost stochastic order. In *The Mathematics of the Uncertain*, pages 33–44. Springer, 2018. 5, 10

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 4

[8] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 3, 4

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

[10] Rotem Dror, Segev Shlomov, and Roi Reichart. Deep dominance - how to properly compare deep neural models. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2773–2785. Association for Computational Linguistics, 2019. 5, 10

[11] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017. 6

[12] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. 2021. 1, 4

[13] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Compact transformers. https://github.com/SHI-Labs/Compact-Transformers, 2023. 4

[14] 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 (CVPR)*, 2016. 4

[15] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019. 3, 4

[16] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 6

[17] Dan Hendrycks, Steven Basart, Mantas Mazeika, Andy Zou, Joseph Kwon, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. Scaling out-of-distribution detection for real-world settings. In *Proceedings of the 39th International Conference on Machine Learning*, pages 8759–8773. PMLR, 2022. 6

[18] 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. 6

[19] Lu Jiang, Di Huang, Mason Liu, and Weilong Yang. Beyond synthetic noise: Deep learning on controlled noisy labels. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4804–4815. PMLR, 2020. 1

[20] S. Kong and D. Ramanan. Opengan: Open-set recognition via open data generation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 793–802, Los Alamitos, CA, USA, 2021. IEEE Computer Society. 5

[21] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009. 1, 3

[22] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015. 4

[23] 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*. Curran Associates, Inc., 2018. 5, 6

[24] 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. 5, 6

[25] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. In *Advances in Neural Information Processing Systems*, pages 21464–21475. Curran Associates, Inc., 2020. 6

[26] Xixi Liu, Yaroslava Lochman, and Christopher Zach. Gen: Pushing the limits of softmax-based out-of-distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23946–23955, 2023. 5, 6

[27] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. 3, 4

[28] Jie Ren, Stanislav Fort, Jeremiah Liu, Abhijit Guha Roy, Shreyas Padhy, and Balaji Lakshminarayanan. A simple fix to mahalanobis distance for improving near-ood detection. In *ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning*, 2021. 6

[29] Chandramouli Shama Sastry and Sageev Oore. Detecting out-of-distribution examples with Gram matrices. In *Proceedings of the 37th International Conference on Machine Learning*, pages 8491–8501. PMLR, 2020. 5, 6

[30] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3, 4

[31] Yue Song, Nicu Sebe, and Wei Wang. Rankfeat: Rank-1 feature removal for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, 2022. 6

[32] Yiyou Sun and Yixuan Li. Dice: Leveraging sparsification for out-of-distribution detection. In *Computer Vision – ECCV 2022*, pages 691–708, Cham, 2022. Springer Nature Switzerland. 5, 6

[33] Yiyou Sun, Chuan Guo, and Yixuan Li. React: Out-of-distribution detection with rectified activations. In *Advances in Neural Information Processing Systems*, pages 144–157. Curran Associates, Inc., 2021. 5, 6

[34] Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. In *Proceedings of the 39th International Conference on Machine Learning*, pages 20827–20840. PMLR, 2022. 6

[35] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Peter Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-mixer: An all-MLP architecture for vision. In *Advances in Neural Information Processing Systems*, 2021. 1, 4

[36] Dennis Ulmer, Christian Hardmeier, and Jes Frellsen. deep-significance-easy and meaningful statistical significance testing in the age of neural networks. *arXiv preprint arXiv:2204.06815*, 2022. 5, 10

[37] H. Wang, Z. Li, L. Feng, and W. Zhang. Vim: Out-of-distribution with virtual-logit matching. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4911–4920, Los Alamitos, CA, USA, 2022. IEEE Computer Society. 6

[38] Jiaheng Wei, Zhaowei Zhu, Hao Cheng, Tongliang Liu, Gang Niu, and Yang Liu. Learning with noisy labels revisited: A study using real-world human annotations. In *International Conference on Learning Representations*, 2022. 1, 3

[39] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2691–2699, 2015. 1, 3

[40] Jingkang Yang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, WENXUAN PENG, Haoqi Wang, Guangyao Chen, Bo Li, Yiyou Sun, Xuefeng Du, Kaiyang Zhou, Wayne Zhang, Dan Hendrycks, Yixuan Li, and Ziwei Liu. OpenOOD: Benchmarking generalized out-of-distribution detection. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. 1

[41] Kentaro Yoshioka. Vision transformers for cifar. `https://github.com/kentaroy47/vision-transformers-cifar10`, 2022. 4

[42] Jinsong Zhang, Qiang Fu, Xu Chen, Lun Du, Zelin Li, Gang Wang, xiaoguang Liu, Shi Han, and Dongmei Zhang. Out-of-distribution detection based on in-distribution data patterns memorization with modern hopfield energy. In *The Eleventh International Conference on Learning Representations*, 2023. 5, 6

[43] Jingyang Zhang, Jingkang Yang, Pengyun Wang, Haoqi Wang, Yueqian Lin, Haoran Zhang, Yiyou Sun, Xuefeng Du, Kaiyang Zhou, Wayne Zhang, Yixuan Li, Ziwei Liu, Yiran Chen, and Hai Li. Openood v1.5: Enhanced benchmark for out-of-distribution detection. *arXiv preprint arXiv:2306.09301*, 2023. 1, 4, 5

[44] 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*, 2017. 4