LEARN WHAT YOU CAN'T LEARN: REGULARIZED ENSEMBLES FOR TRANSDUCTIVE OUT-OF-DISTRIBUTION DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

In order to deploy machine learning models in the wild, it is not only crucial to perform well on in-distribution data, but they should also be able to detect samples that they cannot predict well. Current out-of-distribution (OOD) detection algorithms for neural networks produce unsatisfactory results in many OOD detection scenarios. One of these approaches indicates test points as OOD if an ensemble of classifiers disagrees on them. For settings where the test data is available at training time, we build on this idea and theoretically motivate a method to diversify ensembles by maximizing prediction discrepancy on the entire test set. Regularizing the hypothesis class then induces an ensemble that disagrees on the OOD samples in the test set while agreeing on ID data (RETO). Using early stopping as a regularization technique on neural networks, our experiments show that RETO ensembling outperforms both inductive and transductive state-of-the-art OOD detection methods.

1 INTRODUCTION

Machine learning (ML) systems have become ubiquitous in many applications in the real world - from aiding medical diagnosis (Beede et al., 2020) to making recommendations for the justice system (Angwin et al., 2016). Despite their popularity, ML methods are very brittle when the samples at test time come from a different distribution than at training time. In fact, this is rather likely to be the norm. For example, sampling biases can lead to spurious correlations in the training set (Sagawa et al., 2020), a faulty sensor can produce novel data corruptions (Lu et al., 2019), or new unseen classes can emerge over time, like new diseases (Cao et al., 2020).

In the particular case that the change in distribution satisfies the covariate shift assumption (Shimodaira, 2000), one can use domain adaptation techniques to obtain estimators with good performance on the test set. However, domain adaptation is not always applicable, e.g. when the test set contains new labels or the shift is too strong. In these situations, one can instead aim to identify the inputs that are far from the training set using anomaly detection or out-of-distribution (OOD) detection methods (Mehrotra et al., 2017; Bulusu et al., 2020).

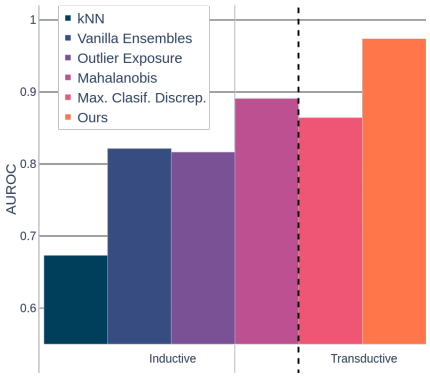


Figure 1: Area under ROC curve (AUROC) averaged over all settings in Table 1.

Recent OOD detection works fall in one of two categories: i) Predictive uncertainty-based or confidence-based approaches want to identify on which samples a given model cannot predict reliably and include Bayesian methods (Gal & Ghahramani, 2016; Malinin & Gales, 2018) or alternatives such as Deep Ensembles (Lakshminarayanan et al., 2017) These methods notably focus on hypothesis classes that achieve good generalization in-distribution. ii) Distribution-based approaches that have the sole goal of identifying samples with

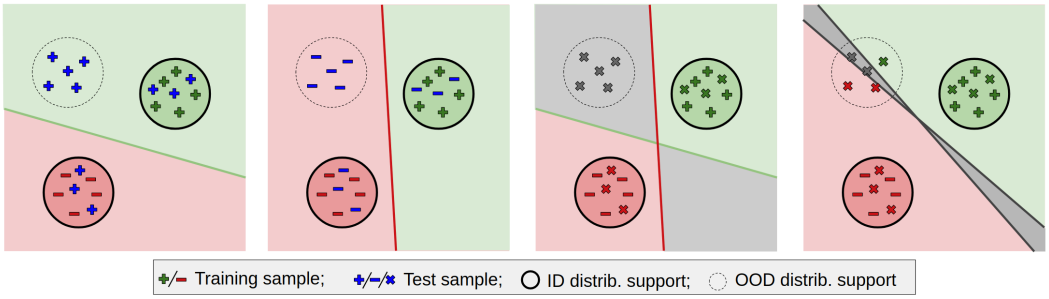


Figure 2: **Left**: Example of a source set labeled according to a true function $f^\star$ and an unlabeled test set with samples both in and outside the support of the source distribution. **Right**: Example of classifiers that agree in-distribution (hatched areas) and disagree on OOD data (gray area). The red and green solid lines are classifiers that fit both the test set labeled with -1 and +1, respectively, and the source set. The signal in the correctly-labeled source does not allow for the arbitrary label to be fit on test ID.

low probability under the training distribution, e.g. using density estimation (Nalisnick et al., 2019) or statistics of the intermediate layers of a neural network (Lee et al., 2018).

Most prior work approaches OOD detection inductively, trying to infer OOD samples after only observing the (ID) training data, but their performance is often poor when outliers are similar to the ID samples (see inductive methods in Figure 1). However, in many practical applications the batch of (unlabeled) test data can actually be available together with the training data, i.e. *transductive* setting (Vapnik, 1998), for instance when real-time prediction is not necessary and computational resources are available for fine-tuning the model parameters at test time. Examples include performing inference offline based on measurements from a sensor network with a novel fault in one of the sensors, or detecting the spread of a new virus: suddenly a lab receives for analysis many MRIs from patients that exhibit the same previously unseen symptoms. Hence, for OOD settings on which inductive methods fail, we explore the following question in this paper:

Is it possible to leverage the test samples in a transductive setting, to obtain ensembles that are diverse (only) outside the support of the training distribution?

We propose using a Regularized Ensemble for Transductive OOD detection (RETO). On a high level, we promote ensemble diversity by encouraging the models to give different predictions on the OOD samples in the test set. However, we do not know which data points are OOD within the test set. Our proposed solution is to train ensembles on both the (labeled) training set and the unlabeled test set for which we assign a uniform label (different for each model in the ensemble). By using strong regularization (e.g. early stopping), the models will fit the arbitrary label *only on the OOD* samples in the test set, but not on the ID samples. Our method not only outperforms state-of-the-art OOD detection baselines on standard OOD benchmarks, it also achievesremarkable performance on difficult image OOD detection scenarios, as shown in Figure 1.

We want to note that in a recent work, Yu & Aizawa (2019) consider a similar transductive setting, trying to obtain two classifiers that maximally disagree on the test set. However, due to the absence of strong regularization, the method produces indistinguishable outputs on ID and OOD samples, as shown in the experiments in Figure 1.

2 Setting and Proposed Method

There are two different ways to define the problem of OOD detection and most recent work on this topic falls in one of these categories. Assuming a classification setting, they are as follows:

1. **Predictive uncertainty-based OOD detection:** Given a labeled training set and a function class, we want to find a function in that class that has simultaneously low error on the training distribution and high uncertainty on inputs far from the support of the training distribution.

2. **OOD detection independent of predictive hypothesis class:** Given samples from a distribution, we want to test whether a test point was drawn from that same distribution.

Our goal in this paper is primarily to warn the practitioner of test samples on which a predictive model's output should not be trusted and hence focus on the first definition. We consider settings where data scientists have already found an algorithm, e.g. a combination of hypothesis class $\mathcal{F}$, loss function and optimization procedure, such that the minimizer generalizes well on ID data.

In this section we formally define predictive uncertainty-based OOD problems that are well-posed and which depend on the relationship between the hypothesis class $\mathcal{F}$ and the support of the training distribution. Formalizing the goal mathematically then directly suggests our method of regularized ensembles for large models such as neural networks.

ID data. We assume that our training set $S := \{(x_i, y_i)\}_{i=1}^n$ consists of samples drawn from the joint distribution P over random variables X, Y in $\mathcal{X} \times \mathcal{Y}$. With a slight abuse of notation, we denote by $\text{supp}_P \subset \mathcal{X}$ the support of the marginal distribution P_X. For simplicity, we consider a noiseless setting, i.e. we assume that there exists a deterministic function $f^\star : \text{supp}_P \to \mathcal{Y}$ such that the predictor variable is given by $y = f^\star(x), \forall x \in \text{supp}_P$.[1]

Transductive OOD detection. In this work we consider a transductive OOD detection setting where the test set is also available during training. It contains a set T_{OOD} (test OOD) that lies outside the support of P_X and a set T_{ID} (test ID) drawn from P_X (hence $T_{\text{ID}} \subset \text{supp}_P$). We further assume that we have a lot more correctly-labeled training data than test data, i.e. $|S| \gg |T_{\text{ID}}|$, to ensure a sufficiently large signal-to-noise ratio.

2.1 Hypothesis class-dependent OOD detection

Recall that our objective is to detect T_{OOD} that the algorithm cannot possibly learn well. This section makes this concept more precise. In particular we answer the question

What are OOD samples that we want to be able to detect?

Given any function class $\mathcal{F}$, we first define the set on which no algorithm searching in $\mathcal{F}$ can can make confident predictions with respect to a training distribution P as the $\mathcal{F}$*-detectable* OOD set.

Definition 2.1 ($\mathcal{F}$-detectable OOD set)**.** *For a given distribution P, we define the $\mathcal{F}$-detectable OOD set $\text{OOD}(P, \mathcal{F})$ as the set of samples on which two or more optimal predictors on P have different outputs:*

$$\text{OOD}(P, \mathcal{F}) := \{x \in \mathcal{X} : \exists f_1, f_2 \text{ s.t. } f_1(x) \neq f_2(x) \wedge f_1, f_2 \in \arg\min_{f \in \mathcal{F}} \mathbb{E}_P \left[\mathbb{I}_{f(x) \neq y}\right]\}$$

An illustration of $\text{OOD}(P, \mathcal{F})$ in a simple 2D example with $\mathcal{F} = \mathcal{F}_{\text{lin}}$ is depicted in Figure 3. The largest detectable set for an arbitrarily large hypothesis class would be the complement of supp_P with respect to $\mathcal{X}$. This situation is however of no interest to us, since such large hypothesis classes do not have good generalization properties when only finite samples are available.

Hypothesis class we care about. What are interesting hypothesis classes $\mathcal{F}$ that we care about? In practice, in our case, we use an algorithm to learn a model within some hypothesis class $\mathcal{F}$ using finite data such that it generalizes well. That is, we obtain a model in $\mathcal{F}_\epsilon^\star := \{f \in \mathcal{F} : \mathbb{E}\left[\mathbb{I}_{f(x) \neq y}\right] < \epsilon\}$. Since in predictive uncertainty-based OOD detection we only care about models that generalize well on the training distribution, what we want to detect are points in $\text{OOD}(P, \mathcal{F}_\epsilon^\star)$. If we knew the entirety of the set $\mathcal{F}_\epsilon^\star$, then we could indeed detect all of $\text{OOD}(P, \mathcal{F}_\epsilon^\star)$.

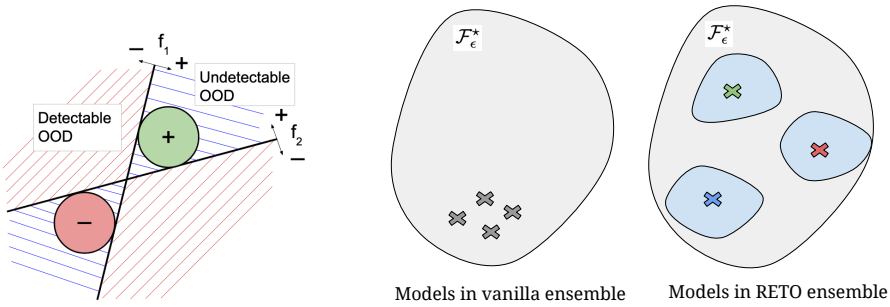
In the transductive setting we care about a specific $T_{\text{OOD}} \subset \text{OOD}(P, \mathcal{F}_\epsilon^\star)$ and we cannot obtain $\mathcal{F}_\epsilon^\star$ when only finite samples are available. How and when can we still detect T_{OOD}?

2.2 RETO: Regularized ensembles for transductive OOD detection

Ensemble-based detection The key is to have a diverse enough ensemble such that for all points in T_{OOD} there exist at least two models that give contradictory predictions.

First, we need to find different models that generalize well, i.e. are in $\mathcal{F}_\epsilon^\star$. We assume that within $\mathcal{F}$, all minimizers of the empirical error $\mathcal{M}(S, \mathcal{F}) := \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \mathbb{I}_{f(x_i) \neq y_i}$ generalize

[1] Note that outside of the support P_X, the function $f^\star$ is undefined, which indicates that domain adaptation is not guaranteed to work for all OOD samples.



(a) (b)

Figure 3: (a) The $\mathcal{F}_{\text{lin}}$-(un)detectable OOD subset. The red and green circles represent the support of the training (ID) distribution supp_P. The **black** solid lines are the decision boundaries of two linear classifiers with zero bias that disagree *maximally* outside supp_P. (b) Cartoon illustration contrasting vanilla and RETO ensembles **Left**: The vanilla ensemble often outputs models that predict similarly on T_{OOD} **Right**: The blue sets here indicate minimizers of the different labelings, i.e. $\mathcal{M}(S \cup T^c, \widetilde{\mathcal{F}})$ for different c. Since the sets are now more separate (as in they label T_{OOD} differently) the models that the algorithm outputs will also be more diverse

well, i.e. $\mathcal{M}(S, \mathcal{F}) \subset \mathcal{F}_\epsilon^\star$. Empirically, this assumption seems to hold true for trained deep neural networks (Theisen et al., 2020). In order to get a diverse set of models, Deep Ensembles (Lakshminarayanan et al., 2017) use the stochasticity in the training process. However the models in the ensembles using this procedure are still very close (see Fig 3 (b)) and tend to agree on T_{OOD}.

Our method aims at generating ensembles that are more diverse on the set T_{OOD}. If we were given $T = T_{\text{OOD}}$, we could just enforce different models to learn different labels on T by simply assigning one of the labels $c \in \mathcal{Y}$ to the whole test set to obtain $T^c := \{(x_i, c) : x_i \in T\}$, and then train our model on $S \cup T^c$. But obviously we are given the union of the test ID and test OOD set $T = T_{\text{OOD}} \cup T_{\text{ID}}$, without being able to distinguish between the two.

How can we encourage an ensemble to disagree (only) on T_{OOD}?

Could we still use the same strategy of assigning an arbitrary label to the entire test set? This maximum discrepancy approach was exactly what Yu & Aizawa (2019) proposed. However, a neural network can easily learn random labels (see e.g. (Zhang et al., 2016)). Hence, if we train a neural network on S, T^c, *the models will also disagree on ID!* How can we enforce models to predict differently on T_{OOD} but *agree* on T_{ID}?

Regularized ensembles The magic word for overparameterized models such as neural networks is: regularization. Regularization reduces the size of the hypothesis class in which the algorithm operates, to a subset $\widetilde{\mathcal{F}} \subset \mathcal{F}$. For our use case, $\widetilde{\mathcal{F}}$ should be small enough that it does not fit wrong labels for T_{ID} but large enough to still fit T_{OOD}. Then we can use a disagreement statistic and flag as OOD all the points in T for which at least two models in the ensemble disagree. The algorithm is summarized in Algorithm 1.

Algorithm 1: Algorithm for RETO

Input: S, T, K, t_0 (the test statistic threshold), $\widetilde{\mathcal{F}}$, disagreement metric
Result: O, i.e. the elements of T which are OOD
for $c \leftarrow \{y_1, y_2, ..., y_K\}$ **do** `// train K models`
  $\hat{f}_c \leftarrow \textit{Train}\left(S \cup T^c; \widetilde{\mathcal{F}}\}\right)$
$O = \emptyset$
for $x \in T$ **do** `// run two-sample test`
  if $\textit{disagreement}\left(\hat{f}_1(x), \ldots, \hat{f}_K(x)\right) > t_0$ **then**
    $O \leftarrow O \cup \{x\}$
return O

We now describe our assumptions on the regularized function class $\widetilde{\mathcal{F}}$ in a bit more detail.

Assumption 2.1 ($\widetilde{\mathcal{F}}$ cannot fit the noisy test ID set)**.** *We define the event:*

$$E_c := \{(S, T_{\text{ID}}) : \mathcal{M}(S \cup T_{\text{ID}}^c, \widetilde{\mathcal{F}}) \subset \mathcal{M}(S, \widetilde{\mathcal{F}}) \subset \mathcal{F}_\epsilon^\star\}.$$

We say that $\widetilde{\mathcal{F}}$ cannot fit the noisy test ID set, if the event E_c has high probability for all $c \in \mathcal{Y}$. [2]

[2]Conditioned on the source set being large enough for the empirical minimizers to generalize well.

Effectively, the function class is small enough and generalizes well enough, that minimizers have to "ignore" wrong labels in the support set of the training distribution. As a consequence of this assumption, fitting $S \cup T_{\text{ID}}$ on $\widetilde{\mathcal{F}}$ with our algorithm only outputs predictors with good generalization which agree on T_{ID}. Note that unrestricted overparameterized models $\mathcal{F}$ do not satisfy Assumption 2.1 due to their capability to fit random labels. However, these are complex non-smooth functions that do not generalize well on P. In Section 3 we provide empirical evidence which shows that Assumption 2.1 holds in many practical scenarios

Now note that using this algorithm, we obtain models from $\mathcal{M}(S, \widetilde{\mathcal{F}})$, which is only an approximation of $\mathcal{F}_\epsilon^\star$, hence obviously we cannot detect all $T_{\text{OOD}} \subset \text{OOD}(P, \mathcal{F}_\epsilon^\star)$. By reducing the size of the function class over which we run optimization, we also limit the set of samples that we can detect as OOD to $\text{OOD}(P, \widetilde{\mathcal{F}}) \subset \text{OOD}(P, \mathcal{F})$. Nonetheless, the good OOD detection performance of our method indicates that using the regularized function class suffices for many hard OOD detection scenarios.

2.3 RETO ON NEURAL NETWORKS VIA EARLY STOPPING

As neural networks are ubiquitous in practice and generalize well with large enough data set, we would like to use them to build regularized ensembles for detecting out-of-distribution samples. As they are capable of interpolating random labels even on the training set, controlling the function class size via regularization is essential. The recent results of Yilmaz & Heckel (2019); Li et al. (2020); Arora et al. (2019) suggest that early stopping helps neural networks be more robust to label noise without sacrificing standard accuracy. Figure 4 shows the learning curves when fitting a neural network on S, T_{ID}^c and T_{OOD}^c for a chosen label c: The training set and test OOD samples are fit first and after epoch 60, the predictor starts fitting the wrong label on the test ID set as well. We can also observe that early stopping at the point when the validation set accuracy (same distribution as training set) drops, captures this phase transition well.

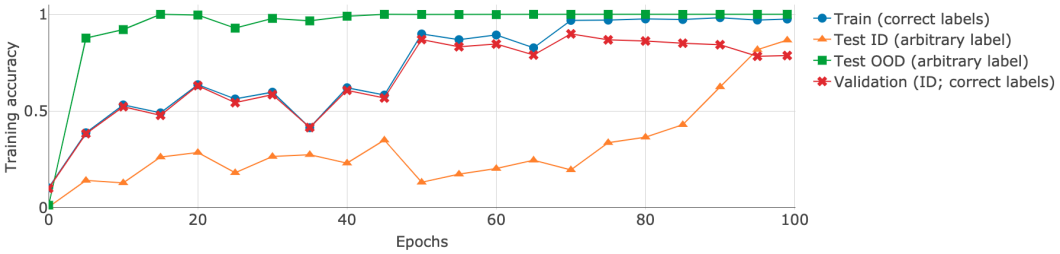


Figure 4: Training accuracy for a ResNet20 measured on the correctly-labeled training set and on the arbitrarily-labeled test ID and test OOD subsets and validation accuracy on a hold-out set with correctly-labeled ID samples. ID samples are from CIFAR10 and OOD samples are from SVHN. The training set and the test OOD samples are fit first, with the test ID set taking longer before it eventually reaches a high accuracy. At the early stopping iteration (after around 50 epochs) the models tend to predict the arbitrary label for test OOD samples and the correct label for test ID.

3 EXPERIMENTS

In this section we evaluate the OOD detection performance of RETO for deep neural networks on several image data sets.

3.1 ID VS. OOD SETTINGS

We consider three categories of OOD detection problems in the order of difficulty: 1) the training and OOD samples come from very different distributions (e.g. CIFAR10 as training, and SVHN as OOD); 2) the OOD samples are "novel" classes (e.g. the first 5 digits of MNIST as training, the last 5 digits as OOD); and 3) the test set contains corrupted samples from the training distribution (e.g. CIFAR10 as training, and CIFAR10-C (Hendrycks & Dietterich, 2019) as OOD) (see Appendix B for more detailed discussion on OOD detection hardness). For CIFAR10-C, we consider corrupted

samples with severity 2 and 5.[3] Note that even though on the corruption data sets, one can potentially use domain adaptation to achieve a classification accuracy of 20.4%[4] (Sun et al., 2020), we use it to provide a strong proof of concept for the potential of RETO to detect hard OOD samples. Appendix D presents more details and examples of images for the various settings.

3.2 BASELINES

We compare our method against both inductive and transductive baselines. Some of the approaches do not assume access to any OOD data during training/validation (e.g. k-Nearest Neighbors, Vanilla Ensembles), while others require a set with OOD data for training (either different from, or the same as the one seen at test time). For example, Outlier Exposure (Hendrycks et al., 2019) uses Tiny-Images for training as the set of outliers, irrespective of the OOD set used for evaluation. On the other hand, the *Mahalanobis* baseline (Lee et al., 2018) uses samples from the same OOD distribution seen at test time, to tune parameters. We also present a transductive version of this approach, referred to as *Mahalanobis-T*, which replaces the OOD training set by a set with both ID and OOD samples.

Statistical test for OOD detection We distinguish between ID and OOD samples using a two-sample statistical test with the following hypotheses: $H_0 : x \in \text{supp}_P$ and $H_1 : x \notin \text{supp}_P$, for a sample x. Each baseline proposes their own choice of the test statistic, which is discussed in detail in Appendix A. For RETO we employ the following test statistic:

$$T_{\text{avg-TV}}(x) := \frac{2}{K(K-1)} \sum_{\substack{i,j=1 \\ i<j}}^{K} d_{\text{TV}}\left(f_i(x), f_j(x)\right)$$

with $f_k(x) \in \mathbb{R}^{|\mathcal{Y}|}$ being the softmax output of the kth model in the ensemble, and d_{TV} the total variation distance. The null hypothesis is accepted for high values of $T_{\text{avg-TV}}$. Appendix H contains more details about the choice of the test statistic.

We use two metrics for evaluation that are common in the OOD detection literature: the area under the ROC curve (AUROC; larger values are better) and the false positive rate at a true positive rate of 95% (FRP@95; smaller values are better). It follows from the way in which the hypothesis test is stated that true positives are OOD samples that are indeed flagged as OOD, while the false positives are the ID samples that are incorrectly predicted as OOD.

3.3 MAIN RESULTS

For our method we train ensembles of five ResNet20 (He et al., 2016) networks (results for other architectures are presented in Appendix G). Each model in the ensemble is early stopped based on its validation loss. For all settings, we used a labeled training set of 40,000 samples, a validation set of 10,000 ID samples and an unlabeled test set of 10000 ID samples and 10000 OOD samples. We present results for training the models from random initializations, and for fine-tuning pretrained models (pretraining is always performed on the source set for 100 epochs). When using pretrained weights, as few as three epochs of fine-tuning are enough on average to achieve the performance that we report, which is a significant cut in computation cost compared to having to train ensembles from scratch. We defer the details regarding training the models to Appendix A.

Table 1 summarizes the main empirical results. For the corruption data sets, the table shows the average of the AUROC and FPR@95 taken over all corruptions (A), and the value for the worst-case setting (W). Appendix G contains a more detailed breakdown of these numbers.

In addition to being successful at identifying OOD samples, our method also maintains a good classification performance on the source distribution. The validation accuracy of the early stopped ensembles, averaged over all settings, is only 1.4% smaller than that of vanilla ensembles.

The evaluation for the scenarios presented in Table 1 is performed on the same test set that was used for training, as usual in transductive learning. In addition to that, the OOD detection performance

[3] Both CIFAR10-C and CIFAR100-C contain 15 types of corruptions, at 5 severity levels.

[4] Average over all CIFAR10-C corruptions of severity 5.

Table 1: Main results. The **best** and **second best** metrics are highlighted for each setting. For the corrupted CIFAR data sets, we report for each severity the average and the worst values, over all corruptions. For all instances, we used a training set of 40,000 labeled samples and a test set of size 20,000 with an equal number of ID and OOD samples.

ID data	OOD data	kNN	Vanilla Ensembles	OE	Mahal.	Mahal-T	MCD	RETO (rand init)	RETO (pretrained)
		Inductive				Transductive			
		AUROC ↑ / FPR@95 ↓							
SVHN	CIFAR10	0.92 / 0.32	0.97 / 0.12	**1.00 / 0.00**	**0.99** / 0.02	**0.99** / 0.05	0.97 / 0.15	**1.00 / 0.01**	**0.99** / 0.03
CIFAR10	SVHN	0.81 / 0.74	0.92 / 0.22	0.97 / 0.11	**0.99** / 0.04	**0.99** / 0.04	**1.00 / 0.02**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	SVHN	0.83 / 0.71	0.84 / 0.52	0.82 / 0.50	**0.98** / 0.10	**0.98 / 0.08**	0.97 / 0.27	**1.00 / 0.00**	**1.00 / 0.00**
SVHN[0:4]	SVHN[5:9]	0.54 / 0.92	0.92 / 0.31	0.85 / 0.48	0.92 / **0.29**	0.91 / 0.37	0.91 / 0.49	**0.96 / 0.20**	**0.94** / 0.34
CIFAR10[0:4]	CIFAR10[5:9]	0.59 / 0.86	0.80 / 0.61	0.82 / 0.59	0.79 / 0.73	0.64 / 0.87	0.69 / 0.75	**0.97 / 0.14**	**0.91 / 0.34**
CIFAR100[0:49]	CIFAR100[50:99]	0.51 / 0.96	0.78 / 0.65	0.74 / 0.69	0.72 / 0.80	0.72 / 0.81	0.70 / 0.74	**0.85 / 0.54**	**0.81 / 0.60**
CIFAR10	CIFAR10-C sev 2 (A)	0.59 / 0.84	0.68 / 0.80	0.70 / 0.80	0.84 / 0.47	0.75 / 0.62	0.82 / 0.50	**0.99 / 0.05**	**0.96 / 0.14**
CIFAR10	CIFAR10-C sev 2 (W)	0.50 / 0.92	0.51 / 0.95	0.52 / 0.94	0.58 / 0.92	0.55 / 0.93	0.52 / 0.94	**0.86 / 0.59**	**0.68 / 0.81**
CIFAR10	CIFAR10-C sev 5 (A)	0.67 / 0.72	0.84 / 0.51	0.86 / 0.46	0.94 / 0.20	0.88 / 0.37	**0.95** / 0.16	**1.00 / 0.00**	**1.00 / 0.01**
CIFAR10	CIFAR10-C sev 5 (W)	0.51 / 0.94	0.60 / 0.90	0.63 / 0.89	0.78 / 0.73	0.68 / 0.88	0.60 / 0.92	**1.00 / 0.00**	**0.98 / 0.14**
CIFAR100	CIFAR100-C sev 2 (A)	0.59 / 0.84	0.68 / 0.80	0.65 / 0.81	0.82 / 0.52	0.67 / 0.78	0.72 / 0.71	**0.97 / 0.14**	**0.94 / 0.24**
CIFAR100	CIFAR100-C sev 2 (W)	0.50 / 0.95	0.52 / 0.94	0.52 / 0.94	0.55 / 0.93	0.55 / 0.94	0.52 / 0.94	**0.86 / 0.56**	**0.71 / 0.81**
CIFAR100	CIFAR100-C sev 5 (A)	0.67 / 0.73	0.78 / 0.63	0.76 / 0.63	0.92 / 0.28	0.84 / 0.45	0.91 / 0.35	**1.00 / 0.01**	**0.99 / 0.03**
CIFAR100	CIFAR100-C sev 5 (W)	0.51 / 0.94	0.64 / 0.86	0.62 / 0.87	0.71 / 0.81	0.63 / 0.87	0.60 / 0.90	**0.98 / 0.11**	**0.96 / 0.29**
	Average	0.67 / 0.77	0.82 / 0.52	0.82 / 0.51	0.89 / 0.34	0.84 / 0.44	0.86 / 0.42	**0.97 / 0.11**	**0.95 / 0.17**

of RETO extrapolates well to unseen samples from the same distribution.[5] In order to show this, we run experiments in which we compute the AUROC on a hold-out test set drawn from the same ID and OOD distributions as the one used during training. The AUROC on the hold-out test set is within 0.01 from the one calculated on the test set observed during training.

Table 2: Results on Tiny ImageNet vs Tiny ObjectNet

ID data	OOD data	kNN	Vanilla Ensembles	OE	RETO (pretrained)
		AUROC ↑ / FPR@95 ↓			
Tiny ImageNet	Tiny ObjectNet	0.51 / 0.96	**0.82 / 0.51**	0.79 / 0.63	**0.98 / 0.12**

Additionally, we also present in Table 2 experiments on ImageNet, where the OOD samples are from ObjectNet (Barbu et al., 2019), a data set that contains both novel classes that do not appear in ImageNet, and images from ImageNet classes, with strong distribution shift (the top-1 accuracy of classifiers trained on ImageNet drops by 40-50% when evaluating on ObjectNet). All images were resized to 32x32.

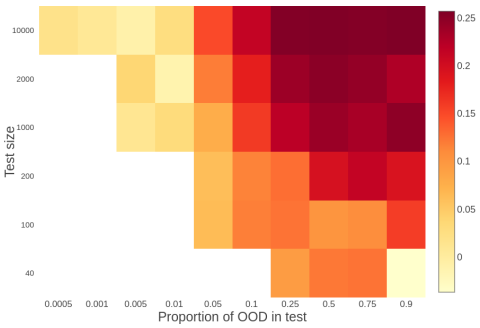


Figure 5: The relative increase in AUROC when using RETO, compared to the vanilla ensemble as a function of the propotion of OOD (CIFAR10-C/snow) vs. ID (CIFAR10) data points in the test set is changed. The AUROCs are obtained with an ensemble of ResNet20 models.

Limitations and trade-offs. Having access to the test set in the transductive setting provides enough information to discriminate well between the ID and the OOD test samples, succeeding when inductive approaches are less effective. In order to bridge the gap between transductive OOD detection and (online) anomaly detection (where only one test sample is needed to make a prediction) we investigate the impact of the size of the test set on the OOD detection performance. In addition, we also vary the ratio of OOD samples in the test set, i.e. $\frac{|T_{\text{OOD}}|}{|T_{\text{ID}}|+|T_{\text{OOD}}|}$. Our findings suggest that there is a broad spectrum of values for which RETO maintains a good performance. In the cases when either the size of the

[5]This setting is similar for instance to the one in the Mahalanobis baseline, where the OOD samples come from the same distribution at training and test time.

test set or the test OOD ratio is small, the OOD detection performance deteriorates to the point where it is comparable to vanilla ensembles, as shown in Figure 5 where we report AUROC averaged for CIFAR-10C/snow severity values 2 and 5. In addition, Appendix E shows the dependence on the ensemble size for RETO and Vanilla Ensemble.

4 RELATED WORK

Transductive learning. Transductive learning (Vapnik, 1998) has been successfully used for practical applications in problems like zero-shot learning (Fu et al., 2015; Kodirov et al., 2015; Wan et al., 2019). Scott & Blanchard (2008) proposed to solve the transductive anomaly detection problem simply by discriminating between the ID and OOD distribution with a constraint on the false positive rate. However, it is difficult to assess the predictive uncertainty of a classifier trained on the source set, using this binary classifier. The setting we consider in which one has access to both a labeled and an unlabeled data set, is reminiscent of the problem of **semi-supervised learning**. Approaches based on self-trained predictors have recently been proposed by Kumar et al. (2020); Chen et al. (2020); Sun et al. (2020) in the context of domain adaptation.

Ensemble diversity. Some limitations of ensemble-based OOD detection approaches have been highlighted in recent years. First, neural networks tend to make overconfident predictions even on test samples far from the training distribution (Hein et al., 2019). Moreover, the stochasticity of conventional NN training approaches (e.g. random initialization, SGD) is not sufficient to obtain ensembles that are diverse enough to give good uncertainty estimates on OOD samples (Maennel & Ţifrea, 2020). Some recent works address this problem by adding explicit regularizers that incentivize model diversity, either in a transductive setting, like MCD (Yu & Aizawa, 2019), or inductively (Bahng et al., 2020), but do not manage to detect OOD samples well in hard scenarios.

Bayesian prediction. One of the important appeals of the Bayesian framework is that it directly provides uncertainty estimates together with the predictions, in the form of a posterior distribution. Approaches like MC-Dropout (Gal & Ghahramani, 2016) or Deep Prior Networks (Malinin & Gales, 2018) have been proposed in the context of OOD detection, but the uncertainty estimates they provide are often inaccurate on OOD samples (Ovadia et al., 2019). Even though Bayesian Neural Networks (Neal, 1996) have seen some progress in recent years (Graves, 2011; Blundell et al., 2015), but sampling efficiently from the posterior over parameters remains an important open problem.

5 CONCLUSIONS

Good predictive uncertainty-based OOD detection is essential in order for classification systems to be deployed in safety-critical environments. We present a method that achieves state-of-the-art performance on a transductive OOD detection setting. Like other predictive uncertainty-based approaches RETO can ultimately be used downstream on problems like active learning or abstaining on OOD samples in combination with selective classification (Geifman & El-Yaniv, 2017). As future work, we propose a more thorough investigation of the influence of the labeling scheme of the test set on the sample complexity of the method, as well as on how much of the $\mathcal{F}^\star$-detectable OOD set can actually be detected.

REFERENCES

Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. 2016.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. volume 97 of *Proceedings of Machine Learning Research*, pp. 322–332, 2019.

Hyojin Bahng, Sanghyuk Chun, Sangdoo Yun, Jaegul Choo, and Seong Joon Oh. Learning de-biased representations with biased representations. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in Neural Information Processing Systems 32*, pp. 9453–9463. 2019.

Emma Beede, Elizabeth Baylor, Fred Hersch, Anna Iurchenko, Lauren Wilcox, Paisan Ruamviboonsuk, and Laura M. Vardoulakis. A human-centered evaluation of a deep learning system deployed in clinics for the detection of diabetic retinopathy. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2020.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32th International Conference on Machine Learning*, 2015.

Saikiran Bulusu, Bhavya Kailkhura, Bo Li, Pramod K. Varshney, and Dawn Song. Anomalous instance detection in deep learning: A survey. *arXiv preprint arXiv:2003.06979*, 2020.

Tianshi Cao, Chin-Wei Huang, David Yu-Tung Hui, and Joseph Paul Cohen. A benchmark of medical out of distribution detection. *arXiv preprint arXiv:2007.04250*, 2020.

Beidi Chen, Weiyang Liu, Zhiding Yu, Jan Kautz, Anshumali Shrivastava, Animesh Garg, and Anima Anandkumar. Angular visual hardness. *arXiv preprint arXiv:1912.02279*, 2019.

Yining Chen, Colin Wei, Ananya Kumar, and Tengyu Ma. Self-training avoids using spurious features under domain shift. *arXiv preprint arXiv:2006.10032*, 2020.

Yanwei Fu, Timothy M. Hospedales, Tao Xiang, and Shaogang Gong. Transductive multi-view zero-shot learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37:2332–2345, 2015.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059, 2016.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems 30*, pp. 4878–4887. 2017.

Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems 24*, pp. 2348–2356. 2011.

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.

Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proceedings of the International Conference on Learning Representations*, 2019.

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *Proceedings of the International Conference on Learning Representations*, 2019.

Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. Unsupervised domain adaptation for zero-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2452–2460, 2015.

Ananya Kumar, Tengyu Ma, and Percy Liang. Understanding self-training for gradual domain adaptation. *arXiv preprint arXiv:2002.11361*, 2020.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6402–6413. Curran Associates, Inc., 2017.

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 31*, pp. 7167–7177. 2018.

Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. Proceedings of Machine Learning Research, pp. 4313–4324, 2020.

Alex X. Lu, Amy X. Lu, Wiebke Schormann, David W. Andrews, and Alan M. Moses. The cells out of sample (COOS) dataset and benchmarks for measuring out-of-sample generalization of image classifiers. *arXiv preprint arXiv:1906.07282*, 2019.

Hartmut Maennel and Alexandru Ţifrea. Fourier networks for uncertainty estimates and out-of-distribution detection. 2020.

Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems 32*, pp. 7047–7058, 2018.

Kishan G. Mehrotra, Chilukuri K. Mohan, and HuaMing Huang. *Anomaly Detection Principles and Algorithms*. Springer Publishing Company, Incorporated, 2017.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *Proceedings of the International Conference on Learning Representations*, 2019.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, 1996.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems 32*, pp. 13991–14002. 2019.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *Proceedings of the International Conference on Learning Representations*, 2020.

Clayton Scott and Gilles Blanchard. Transductive anomaly detection. Technical report, 2008.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244, 2000.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Yu Sun, Xiaolong Wang, Liu Zhuang, John Miller, Moritz Hardt, and Alexei A. Efros. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Ryan Theisen, Jason M Klusowski, and Michael W Mahoney. Good linear classifiers are abundant in the interpolating regime. *arXiv preprint arXiv:2006.12625*, 2020.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

Ziyu Wan, Dongdong Chen, Yan Li, Xingguang Yan, Junge Zhang, Yizhou Yu, and Jing Liao. Transductive zero-shot learning with visual structure constraint. In *Advances in Neural Information Processing Systems 32*, pp. 9972–9982, 2019.

Fatih Furkan Yilmaz and Reinhard Heckel. Image recognition from raw labels collected without annotators. *arXiv preprint arXiv:1910.09055*, 2019.

Qing Yu and Kiyoharu Aizawa. Unsupervised out-of-distribution detection by maximum classifier discrepancy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

A EXPERIMENT DETAILS

A.1 BASELINES

For all baselines, we use pre-trained models provided by the authors when available, and pre-train our own when not available. When doing our own pre-traininig, we always use the parameters presented in the original paper.

- **Vanilla Ensembles**: Train an ensemble on the Source classification problem. For a test sample, average the models' probabilities, and use the entropy of the resulting distribution as the test statistic. We use ensembles of 5 models, with the same architecture and hyperparameters as the ones used for RETO.
- **Binary Classifier**: Train a binary classifier on Source vs Target, and perform early stopping based on validation Source data.
- **Outlier Exposure** Hendrycks et al. (2019): Makes the model's softmax predictions close to the uniform distribution on the outlier distribution, while maintaining good classification performance on the original distribution. We use the WideResNet for the RGB data sets.
- **UOOD MCD** Yu & Aizawa (2019) Train two classifiers at the same time, and makes them disagree on Target data, while maintaining good classification performance.

  For the black and white datases (the MNIST, fashion MNIST and their splits) we resize the images to 32×32 and we make them RGB. We use the WideResNet models.
- **k Nearest Neighbors**, we take $k = 8$. For each Target sample, we take the average distance to the nearest neighbors in the input space, and we use this as the test statistic.
- **Mahalanobis** Lee et al. (2018) pretrains models on the Source data. For a data point, it uses the intermediate representations of each layer as "extracted features". It then performs binary classification using logistic regression using these extracted features. In the original setting, the classification is done on "training" ID vs "training" OOD samples (which are from the same distribution as the test samples). Furthermore, hyperparameter tuning is performed on "validation" ID and OOD data. Because this is very different from the other settings (where we do not have access to samples which are known to be OOD), we propose a transductive alternative. In the transductive alternative, the classification is done on Source vs Target samples, and the hyperparameters are selected only on validation ID data.

  We use the ResNet models. For black and white data sets we follow the same procedure as for **UOOD MCD**.

A.2 TRAINING CONFIGURATION FOR REGULARIZED ENSEMBLES

For MNIST, we train a 3-layer MLP with ReLU activations. Each intermediate layer has 100 neurons. The model is optimized using Adam, using a learning rate of 0.001, for 10 epochs.

For CIFAR, we train a slightly modified ResNet20 model. The kernels are 3×3, and the starting number of filters is 64. The model is trained using SGD with momentum 0.9, and the learning rate starts at 0.1, and is multiplied by 0.2 at epochs 50, 70 and 90. The weights have a l2 regularization of $5e-4$.

For ImageNet we use the same training setting as for CIFAR.

B OOD DETECTION HARDNESS

Consider the task of distinguishing between samples that come from two distributions with disjoint supports $\text{supp}_P, \text{supp}_Q$. Let us assign labels according to the distribution the points are coming from: $\mathcal{D} = \{(x_i, y_i) : y_i = -1 \text{ if } x_i \in \text{supp}_P, y_i = 1 \text{ if } x_i \in \text{supp}_Q\}$.

We define the *hardness* of the OOD detection task with respect to a function class $\mathcal{F}$ as the area under the curve of the training loss. f_0 is the model randomly initialized, and f_t is the model after the fraction t of training is done.

$$\mathcal{H}_{OODD}(\mathcal{D};\mathcal{F}) := \int_0^1 L(f_t)dt$$

For our task, we start with a VGG model and train it for 30 epochs. In order to approximate the integral, we take the training loss for the whole data set every 5 epochs, and we average these losses.

Table 3: Hardness for various tasks.

ID data set	**OOD data set**	$\mathcal{H}_{OODD}$
MNIST	FashionMNIST	0.01
FashionMNIST	MNIST	0.01
MNIST[0:4]	MNIST[5:9]	0.15
FashionMNIST:0,2,3,7,8	FashionMNIST:1,4,5,6,9	0.27
CIFAR10	SVHN	0.05
SVHN	CIFAR10	0.05
ImageNet	ObjectNet	0.18
CIFAR10[0:4]	CIFAR10[5:9]	0.45
SVHN[0:4]	SVHN[5:9]	0.24
CIFAR100	SVHN	0.06
CIFAR100[0:49]	CIFAR100[50:99]	0.59
CIFAR10	CIFAR10-C sev2	0.18
CIFAR10	CIFAR10-C sev5	0.08
CIFAR100	CIFAR100-C sev2	0.19
CIFAR100	CIFAR100-C sev5	0.10

We compare the hardness of some OOD detection tasks that are commonly used in the literature in table 3. For the corrupted CIFAR data sets, we report the average over all corruptions of the given severity.

C REGULARIZED SOURCE/TARGET DISCRIMINATOR FOR TRANSDUCTIVE OOD DETECTION

The paper Scott & Blanchard (2008) suggests that training a binary classifier with bounded false positive rate to distinguish between the source set S and the target set T can successfully separate the OOD samples from the ID samples in the target set. However, this approach does not fall in the category of predictive uncertainty-based OOD detection methods, since it does not provide a good classifier of the labeled training set as well.

We present in what follows a set of experiments run to check if a similar technique works for the data sets we considered. Early stopping with respect to a validation set that contains only ID samples is enough to obtain good OOD detection performance.

D ID AND OOD DATA SETS

D.1 DATA SETS

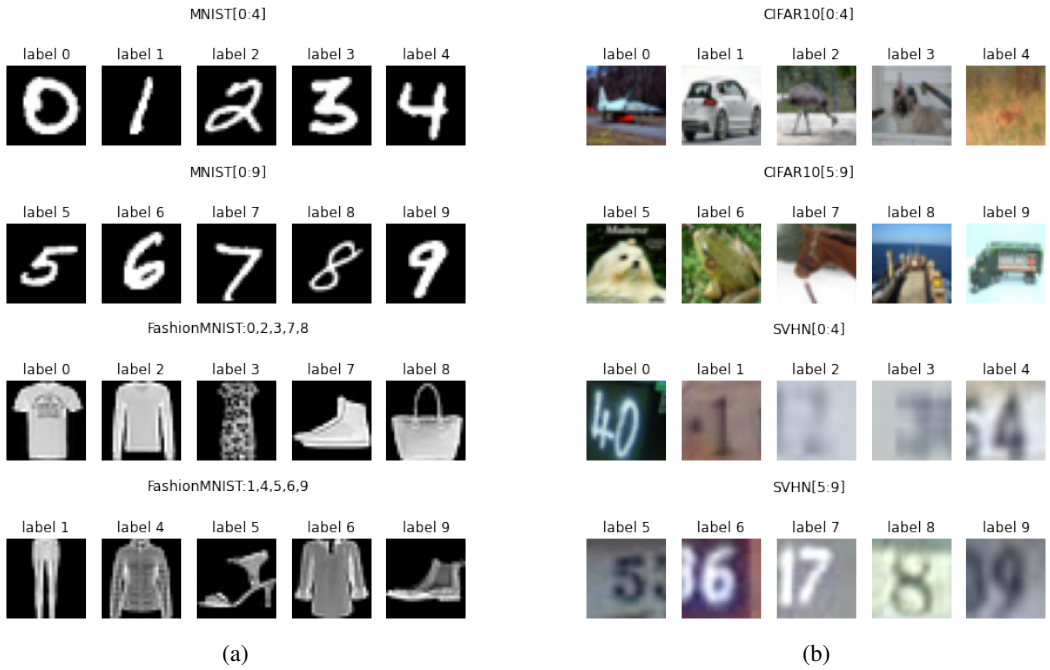
We use:

- MNIST and Fashion MNIST for the easier settings.
- CIFAR10, CIFAR100, their corrupted variants, and SVHN for the harder settings.
- ImageNet and ObjectNet, both resized to 32x32, for the hardest setting.

Table 4: AUROC for an early-stopped binary classifier trained to separate the source set from the target set.

ID data	OOD data	Binary classifier AUROC ↑
CIFAR10	SVHN	1.00
SVHN	CIFAR10	0.99
CIFAR10[0:4]	CIFAR10[5:9]	0.83
SVHN[0:4]	SVHN[5:9]	0.82
CIFAR100	SVHN	1.00
CIFAR100[0:49]	CIFAR100[50:99]	0.61
CIFAR10	CIFAR10-C sev 2 (A)	0.94
CIFAR10	CIFAR10-C sev 2 (W)	0.51
CIFAR10	CIFAR10-C sev 5 (A)	1.00
CIFAR10	CIFAR10-C sev 5 (W)	1.00
CIFAR100	CIFAR100-C sev 2 (A)	0.83
CIFAR100	CIFAR100-C sev 2 (W)	0.50
CIFAR100	CIFAR100-C sev 5 (A)	1.00
CIFAR100	CIFAR100-C sev 5 (W)	1.00
Average		0.94

D.2 SAMPLES FOR THE SETTINGS WITH NOVEL CLASSES



(a) (b)

Figure 6: (a) Data samples for the MNIST/FashionMNIST splits. (b) Data samples for the CIFAR10/SVHN splits.

D.3 SAMPLES FROM OBJECTNET

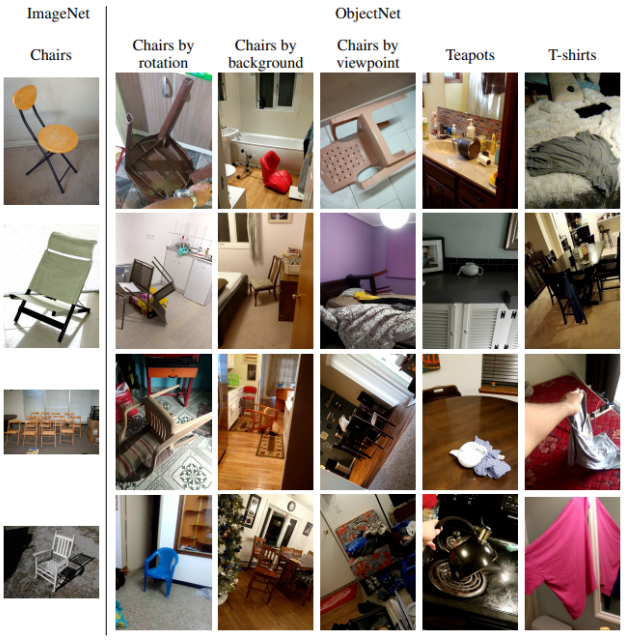


Figure 7: Samples from ImageNet and ObjectNet taken from the original paper by Barbu et al. (2019).

D.4 SAMPLES FROM CIFAR10-C

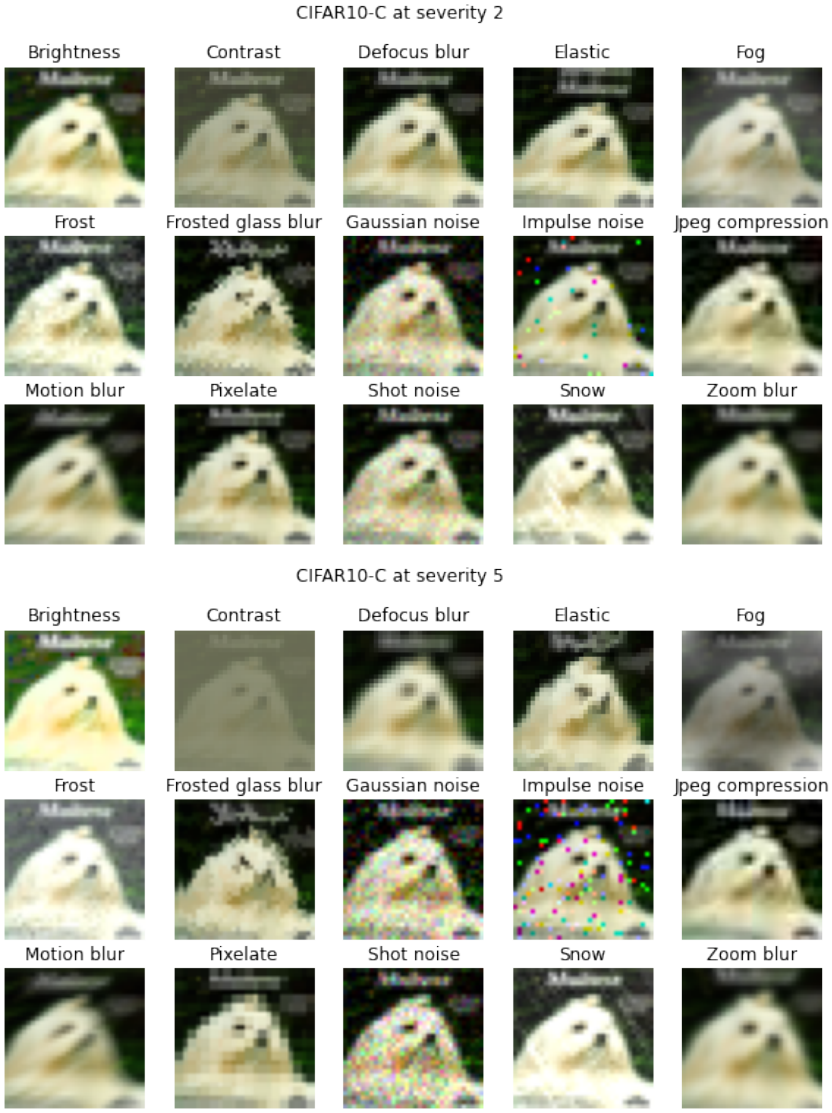


Figure 8: Data samples for the corrupted CIFAR10-C data set.

E RESOURCE REQUIREMENTS FOR RETO

Computational cost Our method can work with training each model in the ensemble from scratch (i.e. random initialization), but it also preforms well when fine-tuning a network from pretrained weights. This reduces significantly the inference time for each batch of test data. For the setting we considered, on average, as few as three epochs of fine tuning are enough to achieve the best performance (the training is stopped early, on average, after three epochs, according to the condition on the validation loss).

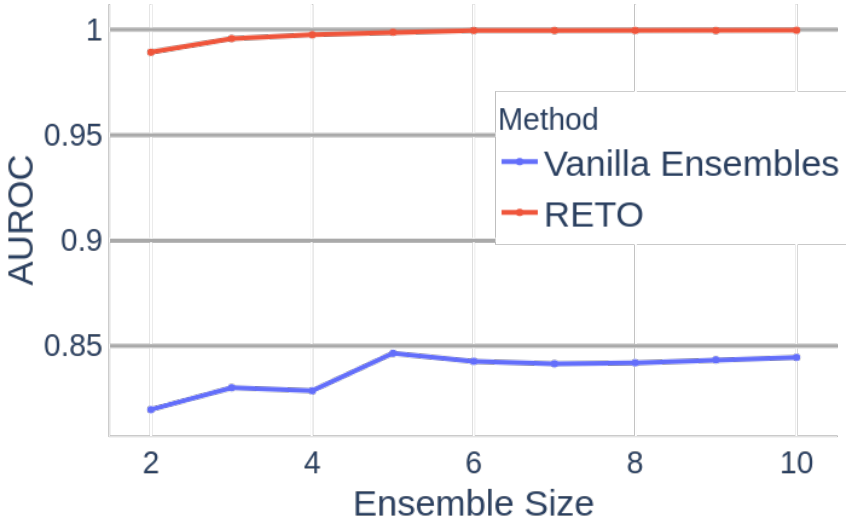


Figure 9: Effect of ensemble size on CIFAR100 vs. SVHN. Both methods are trained from scratch for 100 epochs.

Dependence on ensemble size Figure 9 shows that for Vanilla Ensembles, there is a large increase when going from 2 to 5, whereas for RETO the performance is almost maximal right from the start. This is because, in Vanilla, the networks are diverse 'by chance', due to the stochasticity of the training procedure. On the other hand, in RETO, our training method actively encourages the ensembles to be diverse, and hence two models will already disagree on OOD data almost as much as five.

F GENERALIZATION TO HOLD-OUT TARGET SET

In this section we present experiments which show that after training/fine-tuning on a target set with ID and OOD samples, one can also use our method to detect OOD samples from the same distribution, that have not been seen during training. Concretely, we use a target set of 5000 ID and 5000 OOD samples to train RETO ensembles (we reiterate that we do not have access to which samples are indeed OOD in the target set). For evaluation, we compute the metrics on a separate data set, with 5000 ID and 5000 OOD samples, where the OOD samples come from the same distribution as the samples seen during training. As revealed in Table 5, the performance does not change substantially, when evaluating on the training or on the hold-out target set.

Table 5: Generalization on held out test set.

ID data	OOD data	RETO (normal)	RETO (holdout)
		AUROC ↑ / FPR@95 ↓	
CIFAR10	SVHN	**1.00 / 0.00**	**1.00 / 0.00**
SVHN	CIFAR10	**0.99 / 0.03**	**0.99 / 0.02**
CIFAR10[0:4]	CIFAR10[5:9]	**0.91 / 0.34**	**0.90 / 0.37**
SVHN[0:4]	SVHN[5:9]	**0.94 / 0.34**	**0.95 / 0.28**
CIFAR100	SVHN	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100[0:49]	CIFAR100[50:99]	**0.81 / 0.60**	**0.79 / 0.63**
CIFAR10	CIFAR10-C sev 2 (A)	**0.96 / 0.14**	**0.95 / 0.16**
CIFAR10	CIFAR10-C sev 2 (W)	**0.68 / 0.81**	**0.65 / 0.84**
CIFAR10	CIFAR10-C sev 5 (A)	**1.00 / 0.01**	**1.00 / 0.02**
CIFAR10	CIFAR10-C sev 5 (W)	**0.98 / 0.14**	**0.97 / 0.21**
CIFAR100	CIFAR100-C sev 2 (A)	**0.94 / 0.24**	**0.95 / 0.23**
CIFAR100	CIFAR100-C sev 2 (W)	**0.71 / 0.81**	**0.71 / 0.84**
CIFAR100	CIFAR100-C sev 5 (A)	**0.99 / 0.03**	**0.99 / 0.03**
CIFAR100	CIFAR100-C sev 5 (W)	**0.96 / 0.29**	**0.95 / 0.30**
Average		**0.97 / 0.11**	**0.97 / 0.11**

G MORE EXPERIMENTS

G.1 EXTENDED RESULTS WITH RESNET

Table 6: Extended results on all setting (including a breakdown for all corruption types). For RETO and vanilla ensembles, we train 5 ResNet20 models for each setting.

ID data	OOD data	kNN	Vanilla Ensembles	OE	Mahal.	Mahal-T	MCD	RETO (rand init)	RETO (pretrained)
		AUROC ↑ / FPR@95 ↓							
SVHN	CIFAR10	0.92 / 0.32	0.97 / 0.12	**1.00 / 0.00**	**0.99** / 0.02	**0.99** / 0.05	0.97 / 0.15	**1.00 / 0.01**	**0.99** / 0.03
CIFAR10	SVHN	0.81 / 0.74	0.92 / 0.22	0.97 / 0.11	**0.99** / 0.04	**0.99** / 0.04	**1.00 / 0.02**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	SVHN	0.83 / 0.71	0.84 / 0.52	0.82 / 0.50	**0.98** / 0.10	**0.98 / 0.08**	0.97 / 0.27	**1.00 / 0.00**	**1.00 / 0.00**
SVHN[0:4]	SVHN[5:9]	0.54 / 0.92	0.92 / 0.31	0.85 / 0.48	0.92 / **0.29**	0.91 / 0.37	0.91 / 0.49	**0.96 / 0.20**	**0.94** / 0.34
CIFAR10[0:4]	CIFAR10[5:9]	0.59 / 0.86	0.80 / 0.61	0.82 / 0.59	0.79 / 0.73	0.64 / 0.87	0.69 / 0.75	**0.97 / 0.14**	**0.91 / 0.34**
CIFAR100[0:49]	CIFAR100[50:99]	0.51 / 0.96	0.78 / 0.65	0.74 / 0.69	0.72 / 0.80	0.72 / 0.81	0.70 / 0.74	**0.85 / 0.54**	**0.81 / 0.60**
CIFAR10	CIFAR10-C brightness 2	0.54 / 0.93	0.51 / 0.95	0.52 / 0.94	0.58 / 0.92	0.55 / 0.93	0.52 / 0.94	**0.86 / 0.59**	**0.68 / 0.81**
CIFAR10	CIFAR10-C contrast 2	0.94 / 0.20	0.59 / 0.91	0.62 / 0.89	0.82 / 0.56	0.69 / 0.79	0.82 / 0.91	**1.00 / 0.00**	**0.97 / 0.17**
CIFAR10	CIFAR10-C defocus blur 2	0.56 / 0.91	0.54 / 0.93	0.56 / 0.93	0.67 / 0.81	0.58 / 0.90	0.60 / 0.90	**0.99 / 0.07**	**0.94 / 0.32**
CIFAR10	CIFAR10-C elastic 2	0.56 / 0.92	0.63 / 0.87	0.65 / 0.87	0.71 / 0.76	0.66 / 0.85	0.68 / 0.91	**0.99 / 0.08**	**0.93 / 0.37**
CIFAR10	CIFAR10-C fog 2	0.81 / 0.52	0.53 / 0.93	0.55 / 0.93	0.81 / 0.62	0.71 / 0.73	0.58 / 0.92	**0.99 / 0.06**	**0.90 / 0.39**
CIFAR10	CIFAR10-C frost 2	0.50 / 0.92	0.69 / 0.84	0.69 / 0.85	**0.90** / 0.42	0.72 / 0.73	0.59 / 0.95	**1.00 / 0.00**	**1.00 / 0.01**
CIFAR10	CIFAR10-C frosted glass blur 2	0.51 / 0.95	0.89 / 0.42	0.86 / 0.50	**0.99 / 0.02**	**0.99** / 0.04	**0.99** / 0.04	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C gaussian noise 2	0.57 / 0.89	0.83 / 0.62	0.85 / 0.58	**1.00 / 0.01**	**0.99** / 0.02	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C impulse noise 2	0.61 / 0.85	0.80 / 0.68	0.81 / 0.61	**1.00 / 0.02**	**0.99** / 0.04	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C jpeg compression 2	0.51 / 0.95	0.71 / 0.81	0.73 / 0.77	**0.75 / 0.74**	0.68 / 0.84	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C motion blur 2	0.60 / 0.88	0.71 / 0.81	0.74 / 0.77	0.80 / 0.69	0.79 / 0.61	**0.99 / 0.01**	**1.00 / 0.00**	**0.99** / 0.02
CIFAR10	CIFAR10-C pixelate 2	0.52 / 0.94	0.65 / 0.88	0.66 / 0.85	0.79 / 0.71	0.66 / 0.86	0.93 / 0.73	**1.00 / 0.00**	**0.99 / 0.04**
CIFAR10	CIFAR10-C shot noise 2	0.53 / 0.92	0.74 / 0.79	0.75 / 0.82	**0.98 / 0.06**	0.80 / 0.57	0.96 / 0.23	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C snow 2	0.57 / 0.90	0.74 / 0.76	0.74 / 0.80	0.96 / 0.18	0.73 / 0.74	0.69 / 0.95	**1.00 / 0.00**	**0.99 / 0.01**
CIFAR10	CIFAR10-C zoom blur 2	0.59 / 0.89	0.71 / 0.77	0.73 / 0.81	**0.83 / 0.56**	0.77 / 0.66	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C brightness 5	0.56 / 0.88	0.60 / 0.90	0.63 / 0.89	0.78 / 0.73	0.79 / 0.72	0.60 / 0.92	**1.00 / 0.00**	**0.98 / 0.14**
CIFAR10	CIFAR10-C contrast 5	**1.00 / 0.00**	0.92 / 0.25	0.96 / 0.14	**1.00 / 0.01**	**0.99** / 0.04	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C defocus blur 5	0.65 / 0.82	0.88 / 0.39	0.92 / 0.25	**0.96** / 0.18	0.95 / **0.14**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C elastic 5	0.59 / 0.89	0.80 / 0.64	0.78 / 0.74	0.84 / 0.61	0.75 / 0.77	**0.92 / 0.55**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C fog 5	0.89 / 0.27	0.79 / 0.69	0.80 / 0.69	**0.98 / 0.10**	0.78 / 0.61	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C frost 5	0.70 / 0.68	0.84 / 0.59	0.84 / 0.60	0.98 / 0.07	0.79 / 0.61	**0.99 / 0.04**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C frosted glass blur 5	0.55 / 0.92	0.90 / 0.37	0.89 / 0.39	**0.99 / 0.05**	0.98 / 0.07	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C gaussian noise 5	0.67 / 0.75	0.91 / 0.30	**0.96 / 0.17**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C impulse noise 5	0.83 / 0.50	0.94 / 0.17	**0.98 / 0.09**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C jpeg compression 5	0.51 / 0.94	0.79 / 0.68	0.81 / 0.63	**0.82 / 0.61**	0.68 / 0.88	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C motion blur 5	0.66 / 0.82	0.84 / 0.59	0.86 / 0.50	**0.91 / 0.27**	0.86 / 0.45	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C pixelate 5	0.56 / 0.92	0.84 / 0.59	0.88 / 0.49	0.98 / **0.04**	0.97 / 0.12	**0.99** / 0.07	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C shot noise 5	0.66 / 0.77	0.91 / 0.30	**0.95 / 0.20**	**1.00 / 0.00**	**1.00 / 0.01**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C snow 5	0.61 / 0.84	0.77 / 0.70	0.79 / 0.72	**0.95 / 0.25**	0.72 / 0.74	0.82 / 0.80	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR10	CIFAR10-C zoom blur 5	0.63 / 0.84	0.87 / 0.48	0.88 / 0.43	**0.96 / 0.12**	0.91 / 0.30	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	CIFAR100-C brightness 2	0.53 / 0.94	0.52 / 0.94	0.52 / 0.94	0.55 / 0.93	0.55 / 0.94	0.52 / 0.95	**0.86 / 0.56**	**0.71 / 0.81**
CIFAR100	CIFAR100-C contrast 2	**0.94 / 0.21**	0.60 / 0.90	0.58 / 0.90	0.75 / 0.71	0.66 / 0.83	0.60 / 0.91	**0.96 / 0.22**	0.92 / 0.48
CIFAR100	CIFAR100-C defocus blur 2	0.56 / 0.92	0.55 / 0.92	0.54 / 0.93	0.70 / 0.82	0.69 / 0.80	0.52 / 0.94	**0.91 / 0.48**	**0.88 / 0.60**
CIFAR100	CIFAR100-C elastic 2	0.55 / 0.93	0.61 / 0.88	0.59 / 0.89	0.68 / 0.86	0.66 / 0.85	0.56 / 0.92	**0.93 / 0.43**	**0.87 / 0.60**
CIFAR100	CIFAR100-C fog 2	0.82 / **0.52**	0.55 / 0.93	0.53 / 0.93	0.75 / 0.72	0.73 / 0.73	0.54 / 0.93	**0.93 / 0.33**	**0.85** / 0.64
CIFAR100	CIFAR100-C frost 2	0.53 / 0.90	0.71 / 0.79	0.65 / 0.83	0.82 / 0.65	0.71 / 0.75	0.66 / 0.89	**0.98 / 0.09**	**0.99 / 0.03**
CIFAR100	CIFAR100-C frosted glass blur 2	0.50 / 0.95	0.79 / 0.61	0.72 / 0.72	**0.98 / 0.04**	0.96 / 0.17	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	CIFAR100-C gaussian noise 2	0.56 / 0.89	0.81 / 0.59	0.79 / 0.56	**0.99** / 0.03	0.68 / 0.80	**0.99 / 0.01**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	CIFAR100-C impulse noise 2	0.61 / 0.85	0.82 / 0.59	0.83 / 0.55	**0.99 / 0.03**	0.69 / 0.78	**0.99 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	CIFAR100-C jpeg compression 2	0.51 / 0.94	0.70 / 0.79	0.69 / 0.79	0.71 / 0.82	0.66 / 0.89	0.70 / 0.88	**1.00 / 0.01**	**0.98 / 0.08**
CIFAR100	CIFAR100-C motion blur 2	0.60 / 0.89	0.68 / 0.82	0.63 / 0.85	0.82 / 0.57	0.67 / 0.79	0.68 / 0.89	**1.00 / 0.00**	**0.97 / 0.17**
CIFAR100	CIFAR100-C pixelate 2	0.52 / 0.94	0.63 / 0.88	0.61 / 0.87	0.87 / 0.49	0.57 / 0.91	0.77 / 0.89	**1.00 / 0.00**	**0.98 / 0.11**
CIFAR100	CIFAR100-C shot noise 2	0.53 / 0.92	0.75 / 0.73	0.73 / 0.72	**0.94 / 0.23**	0.60 / 0.85	0.83 / 0.79	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	CIFAR100-C snow 2	0.55 / 0.91	0.73 / 0.77	0.69 / 0.80	**0.93** / 0.36	0.61 / 0.88	0.79 / 0.81	**0.99 / 0.02**	**0.99 / 0.05**
CIFAR100	CIFAR100-C zoom blur 2	0.59 / 0.90	0.69 / 0.81	0.63 / 0.85	0.83 / 0.46	0.68 / 0.81	0.71 / 0.90	**1.00 / 0.00**	**0.99 / 0.06**
CIFAR100	CIFAR100-C brightness 5	0.53 / 0.93	0.64 / 0.86	0.62 / 0.87	0.74 / 0.76	0.72 / 0.77	0.60 / 0.90	**0.98 / 0.11**	**0.96 / 0.29**
CIFAR100	CIFAR100-C contrast 5	**1.00 / 0.01**	0.78 / 0.57	0.82 / 0.53	**0.99** / 0.03	0.98 / 0.12	**0.99 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	CIFAR100-C defocus blur 5	0.65 / 0.84	0.81 / 0.63	0.75 / 0.67	0.96 / 0.15	0.95 / 0.19	**0.99 / 0.01**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	CIFAR100-C elastic 5	0.59 / 0.90	0.72 / 0.76	0.69 / 0.79	0.71 / 0.81	0.71 / 0.82	0.63 / 0.85	**1.00 / 0.00**	**0.99 / 0.07**
CIFAR100	CIFAR100-C fog 5	0.90 / 0.25	0.75 / 0.70	0.70 / 0.73	0.92 / 0.40	0.80 / 0.67	0.95 / 0.46	**1.00 / 0.00**	**0.99 / 0.01**
CIFAR100	CIFAR100-C frost 5	0.73 / 0.68	0.79 / 0.63	0.72 / 0.71	0.94 / **0.26**	0.66 / 0.77	**0.95** / 0.45	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	CIFAR100-C frosted glass blur 5	0.55 / 0.93	0.80 / 0.60	0.73 / 0.72	**0.97 / 0.08**	0.76 / 0.70	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	CIFAR100-C gaussian noise 5	0.67 / 0.77	0.84 / 0.42	**0.85 / 0.41**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	CIFAR100-C impulse noise 5	0.84 / 0.48	0.86 / 0.42	**0.96 / 0.20**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	CIFAR100-C jpeg compression 5	0.51 / 0.94	0.77 / 0.71	0.75 / 0.69	0.82 / 0.61	0.73 / 0.82	0.94 / 0.50	**1.00 / 0.00**	**0.99 / 0.04**
CIFAR100	CIFAR100-C motion blur 5	0.66 / 0.83	0.76 / 0.70	0.71 / 0.74	0.91 / 0.29	0.77 / 0.64	0.87 / 0.71	**1.00 / 0.01**	**0.99 / 0.03**
CIFAR100	CIFAR100-C pixelate 5	0.55 / 0.92	0.80 / 0.59	0.80 / 0.58	**0.99 / 0.01**	0.98 / 0.08	**0.99 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	CIFAR100-C shot noise 5	0.66 / 0.79	0.83 / 0.47	0.84 / 0.44	**1.00 / 0.01**	**1.00 / 0.01**	**0.99 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
CIFAR100	CIFAR100-C snow 5	0.56 / 0.90	0.77 / 0.70	0.73 / 0.73	**0.90** / 0.45	0.63 / 0.87	0.79 / 0.77	**0.99 / 0.06**	**0.99 / 0.02**
CIFAR100	CIFAR100-C zoom blur 5	0.64 / 0.85	0.79 / 0.66	0.74 / 0.70	0.92 / 0.29	**0.93 / 0.24**	0.92 / 0.62	**1.00 / 0.00**	**1.00 / 0.00**
	Average	0.64 / 0.79	0.76 / 0.64	0.76 / 0.64	0.88 / 0.37	0.79 / 0.53	0.85 / 0.41	**0.98 / 0.08**	**0.96 / 0.15**

G.2 RESULTS WITH VGG

Table 7: Results on all setting where we used ensembles of 5 VGG16 models for RETO and vanilla.

ID data	OOD data	kNN	Vanilla Ensembles	OE	Mahal.	Mahal-T	MCD	RETO (rand init)	RETO (pretrained)
		AUROC ↑ / FPR@95 ↓							
SVHN	CIFAR10	0.92 / 0.32	0.97 / 0.12	**1.00 / 0.00**	**0.99 / 0.02**	**0.99** / 0.05	0.97 / 0.15	**0.99 / 0.02**	**0.99** / 0.06
CIFAR10	SVHN	0.81 / 0.74	0.88 / 0.31	0.97 / 0.11	**0.99** / 0.04	**0.99** / 0.04	**1.00 / 0.02**	**1.00 / 0.00**	**1.00 / 0.00**
SVHN[0:4]	SVHN[5:9]	0.54 / 0.92	0.89 / 0.40	0.85 / 0.48	0.92 / **0.29**	0.91 / 0.37	0.91 / 0.49	**0.94 / 0.34**	**0.93** / 0.37
CIFAR10[0:4]	CIFAR10[5:9]	0.59 / 0.86	0.74 / 0.71	0.82 / 0.59	0.79 / 0.73	0.64 / 0.87	0.69 / 0.75	**0.95 / 0.26**	**0.91 / 0.37**
CIFAR10	CIFAR10-C sev 2 (A)	0.59 / 0.84	0.66 / 0.83	0.70 / 0.80	0.84 / 0.47	0.75 / 0.62	0.82 / 0.50	**0.98 / 0.07**	**0.94 / 0.21**
CIFAR10	CIFAR10-C sev 2 (W)	0.50 / 0.92	0.51 / 0.95	0.52 / 0.94	0.58 / 0.92	0.55 / 0.93	0.52 / 0.94	**0.77 / 0.67**	**0.68 / 0.81**
CIFAR10	CIFAR10-C sev 5 (A)	0.67 / 0.72	0.80 / 0.59	0.86 / 0.46	0.94 / 0.20	0.88 / 0.37	0.95 / 0.16	**1.00 / 0.00**	**0.99 / 0.04**
CIFAR10	CIFAR10-C sev 5 (W)	0.51 / 0.94	0.58 / 0.90	0.63 / 0.89	0.78 / 0.73	0.68 / 0.88	0.60 / 0.92	**0.99 / 0.01**	**0.95 / 0.28**
Average		0.69 / 0.73	0.82 / 0.49	0.87 / 0.41	0.91 / 0.29	0.86 / 0.39	0.89 / 0.35	**0.98 / 0.11**	**0.96 / 0.17**

G.3 RESULTS ON MNIST AND FASHIONMNIST

Table 8: Results on MNIST/FashionMNIST settings. For RETO and vanilla ensembles, we train 5 3-hidden layer MLP models for each setting.

ID data	OOD data	kNN	Vanilla Ensembles	OE	Mahal.	Mahal-T	MCD	RETO (rand init)	RETO (pretrained)
		AUROC ↑ / FPR@95 ↓							
MNIST	FashionMNIST	**0.98** / 0.10	0.81 / 0.99	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.02**	**1.00 / 0.00**	**1.00 / 0.00**
FashionMNIST	MNIST	0.98 / 0.09	0.87 / 0.58	0.68 / 0.84	**0.99 / 0.03**	**0.99** / 0.04	**1.00 / 0.00**	**1.00 / 0.00**	**1.00 / 0.00**
MNIST[0:4]	MNIST[5:9]	0.90 / 0.42	0.94 / 0.28	0.95 / **0.22**	**0.99 / 0.02**	**0.99 / 0.02**	0.96 / 0.24	**1.00 / 0.02**	**0.99 / 0.02**
FashionMNIST[0,2,3,7,8]	FashionMNIST[1,4,5,6,9]	0.74 / 0.67	0.64 / 0.93	0.66 / 0.88	0.77 / 0.80	0.82 / 0.61	0.78 / 0.70	**0.95 / 0.27**	**0.93 / 0.38**
Average		0.90 / 0.32	0.82 / 0.70	0.82 / 0.49	0.94 / 0.21	0.95 / 0.17	0.94 / 0.24	**0.99 / 0.07**	**0.98 / 0.10**

For FashionMNIST we chose this particular split (i.e. classes 0,2,3,7,8 vs classes 1,4,5,6,9) because it is more difficult for OOD detection than the 0-4 vs 5-9 split.

G.4 MORE RESULTS FOR OUTLIER EXPOSURE

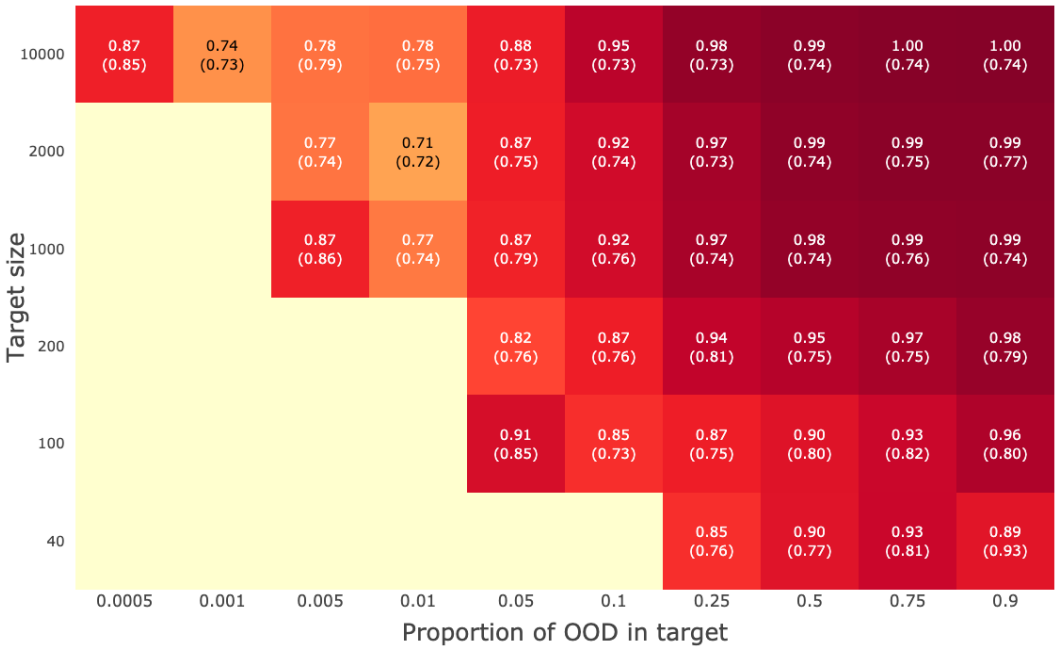
The Outlier Exposure method needs access to a set of OOD samples during training. The numbers we report in the rest of paper for Outlier Exposure are obtained by using the TinyImages data set as the OOD samples that are seen during training. In this section we explore the use of an $\text{OOD}_{\text{train}}$ data set that is more similar to the OOD data observed at test time. This is a much easier setting for the Outlier Exposure method: the closer $\text{OOD}_{\text{train}}$ is to OOD_{test}, the easier it will be for the model tuned on $\text{OOD}_{\text{train}}$ to detect the test OOD samples.

In the table below we focus only on the settings with corruptions. For each corruption type, we use the lower severity corruption as $\text{OOD}_{\text{train}}$ and evaluate on the higher severity data and vice versa.

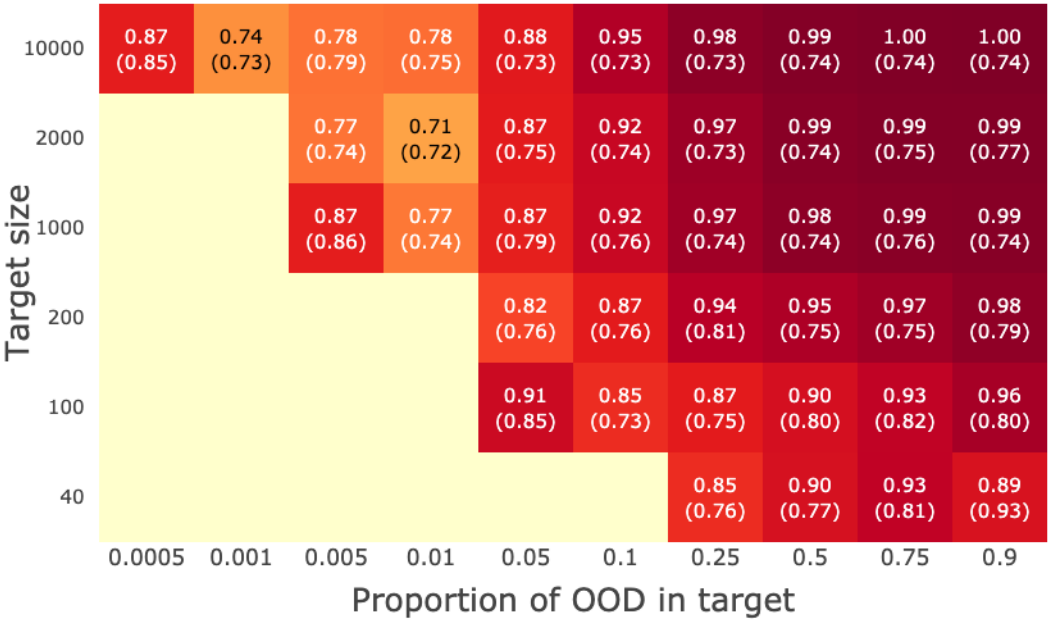
Table 9: Results for Outlier Exposure, when using the same corruption type, but with a higher/lower severity, as OOD data seen during training.

ID data	OOD data	OE (trained on sev5)	OE (trained on sev2)
		AUROC ↑	
CIFAR10	CIFAR10-C sev 2 (A)	0.89	N/A
CIFAR10	CIFAR10-C sev 2 (W)	0.65	N/A
CIFAR10	CIFAR10-C sev 5 (A)	N/A	0.98
CIFAR10	CIFAR10-C sev 5 (W)	N/A	0.78
CIFAR100	CIFAR100-C sev 2 (A)	0.85	N/A
CIFAR100	CIFAR100-C sev 2 (W)	0.59	N/A
CIFAR100	CIFAR100-C sev 5 (A)	N/A	0.97
CIFAR100	CIFAR100-C sev 5 (W)	N/A	0.67
Average		0.87	0.98

G.5 DEPENDENCE ON THE TARGET CONFIGURATION



(a) Initialization from pretrained weights.



(b) Initialization from random weights.

Figure 10: AUROCs obtained with an ensemble of ResNet20 models as the composition of the target set is changed. Only settings with at least 5 OOD samples have been considered. The ID samples are from CIFAR10, while the OOD samples are from CIFAR10-C/snow (AUROC averaged for severity values 2 and 5). For comparison, we provide in parentheses AUROC values for a vanilla ensemble trained once on the source set and evaluated on each target set configuration. In 10a, the models are initialized from pretrained weights and fine tuned. In 10b, the models are initialized with random weights and trained from scratch.

H TWO-SAMPLE TEST FOR OOD DETECTION

Distinguishing between ID and OOD samples can be cast as a two-sample hypothesis test, with $H_0 : x \in \text{supp}_P$ and $H_1 : x \notin \text{supp}_P$, for a sample x. The various OOD detection methods differ in their choice of the test statistic. For approaches that use ensembles of classifiers, the test statistic should reflect the belief that the models have similar outputs on ID samples, and disagree on OOD samples. For example, Lakshminarayanan et al. (2017) propose averaging the softmax outputs of all the models in the ensemble and then taking the maximum or the entropy of the resulting probability vector as the test statistic. For a K-model ensemble and an input x this can be written as follows:

$$T_{\text{max-p}}(x) := \max_{i \in [C]} \frac{1}{K} \sum_{k=1}^{K} \left(f_k(x)\right)_i, \text{ with } f_k(x) \in \mathbb{R}^C \text{ the softmax output of the k}^{\text{th}} \text{ model}$$

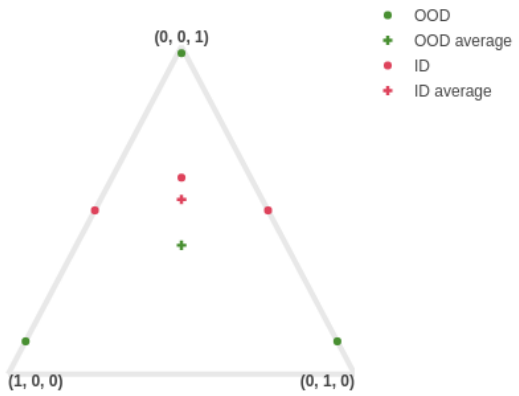


Figure 11: Averaging the probability outputs of the models in an ensemble can lead to catastrophic information loss, in some cases. The softmax vectors of a 3-model ensemble are represented on the 2D probability simplex. For an OOD sample, each model predicts with high confidence the arbitrary label it has seen during training. For the ID sample, the models predict the correct class with moderate confidence. Therefore, the average probability vectors for the ID and the OOD sample are close, which can make it hard to distinguish between them.

Averaging the softmax vectors loses some information about the model predictions, because different initial probability vectors can map to the same averaged vector. In our approach, models are more uncertain on ID samples than on OOD samples, which can make the averaged softmax vector fall at the same location for an ID and an OOD sample. This makes it impossible to distinguish between the two, as illustrated in Figure 11.

For neural network ensembles, following a standard training procedure of minimizing the cross-entropy loss leads to models that make confident predictions on both ID and OOD samples, as shown by Hein et al. (2019); Maennel & Ţifrea (2020). Consequently, the information lost through averaging is not causing any issues: on ID samples, the models will tend to give the same prediction, while on OOD samples the models tend to disagree, giving different predictions with high confidence.

However, in our case, because of early stopping, the training process is halted at different stages for target ID and target OOD samples, as indicated in Figure 4.

Recent papers like Shwartz-Ziv & Tishby (2017); Chen et al. (2019) analyze the dynamics of optimizing the cross-entropy loss with SGD. They suggest that there might exist two stages: one in which a good decision boundary is found, and another in which the margin is increased between the representations of inputs from different classes. It is this second stage that also leads to overconfident predictions on both ID and OOD samples. Thus, early stopping may cause the models to be more uncertain on target ID samples than on target OOD. This is indeed confirmed in Figure 12.

To avoid the problem of information loss described previously, we compute the pairwise total variation distances between the softmax outputs of the models in the ensemble, and we take the average of these distances as our test statistic:

Figure 12: Distribution of confidence for a model trained on the source set alone (**Top**) and a model trained on source and the arbitrarily labeled target: **Middle**: with early stopping and **Bottom**: after 100 epochs. The models in the vanilla ensemble are confident on both ID and OOD samples. The model trained on source+target is only confident on OOD data early during training.

$$T_{\text{avg-TV}}(x) := \frac{2}{K(K-1)} \sum_{\substack{i,j=1 \\ i<j}}^{K} d_{\text{TV}}\left(f_i(x), f_j(x)\right)$$