Benchmarking deep networks for out-of-distribution detection

Ben Cottier

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2020

Abstract

Deployed neural network models may encounter data from an unfamiliar distribution, resulting in poor decisions. One strategy to handle this Out-of-Distribution (OOD) data better is to simply detect it and defer to the user. In this thesis we focus on the OOD detection problem for deep convolutional image classifiers. Specifically, we design a benchmark for detecting OOD images and evaluate several methods. The benchmark combines best practices from previous work, using two kinds of detection task, three training datasets, diverse sources of OOD data, fair constraints on hyperparameter tuning, and comprehensive metrics including runtime. We also explore a new method based on an architecture called TwinNet. The TwinNet method aggregates predictions of partly random neural networks, each trained in under 1/5th the time of a plain neural network. While early experiments indicated some advantages to the TwinNet method, overall it performed worse than the baseline neural network. We therefore do not recommend this new method but believe that further investigation would be valuable. The most effective method trained a classifier to predict image rotations as well as classes. However, all methods degraded severely with the scale of training data. We show that there is still much room for improvement on each of our OOD detection tasks. Our benchmark is therefore suitable for future work that might close the gap.

Acknowledgements

I thank my supervisor Prof. Amos Storkey for supporting this project from inception to completion. His generous review and discussion greatly helped the project and kept me grounded when I got carried away. The idea for the TwinNet model architecture belongs to Amos, with credit to Dr Joseph Mellor for the initial experiments that inspired my own. Joseph also influenced the design of my experiments through helpful discussion and feedback. The experiments were not possible without the generous computing resources provided by Amos and the School of Informatics. I also thank members of the Bayesian and Neural Systems Group, who provided interesting ideas that influenced this project, as well as morale in the midst of a pandemic. Finally, I thank all of the teachers and mentors I had during the MSc program, without whom I would lack the skills and knowledge to accomplish this work.

Table of Contents

Chapter 1

Introduction

When a machine learning model is applied in a real-world setting, it may not be equipped to handle all of the data it encounters. A model to recognise natural objects may see an unknown class of object, or a drawing of a known class. In medical screening, a diagnostic model may encounter a pathology that it was not trained to diagnose, or in autonomous driving, a semantic segmentation model may encounter an unfamiliar obstruction on the road.

Detecting these Out-of-Distribution (OOD) examples is important to make a model trustworthy. This is especially so in safety-critical applications including medical screening and autonomous driving. Through detection, a model can prevent poor decisions by becoming more conservative or deferring to an overseer.

For probabilistic models we might use predictive confidence to detect OOD data. However, predictive confidence is not reliable in regular deep neural networks, as they tend to make overconfident predictions (Guo et al. 2017). Worse, overconfidence persists on OOD data (Hein, Andriushchenko, and Bitterwolf 2019). The problem with overconfidence is that errors may not be obvious. Poor uncertainty estimation can not only lead to actual costly failure when failure is not anticipated, but prevent deployment in the first place due to lack of trust. Either way, poor uncertainty estimates bear an economic and social cost.

Recent work in *out-of-distribution detection* (Hendrycks and Gimpel 2016) is motivated by the inappropriate response of deep neural networks to OOD examples. However, there are several limitations to evaluation. Limitations include evaluating only one training dataset, lack of diversity in OOD datasets, using the same OOD distribution in hyperparameter tuning and testing, and neglecting efficiency as a measure of performance.

This thesis addresses problems in existing evaluations with a new OOD detection benchmark. We design a meaningful set of evaluation tasks, enforce fair standards for hyperparameter tuning, evaluate multiple training datasets, use diverse OOD datasets, and produce a more comprehensive range of metrics than previous work. In addition, we investigate a novel OOD detection method. The new method aggregates the predictions of partially randomised neural networks that can be trained in under 1/5th the time of a plain neural network. The key contributions of the thesis are:

1. A benchmark to evaluate deep neural network methods for OOD detection, which includes a novel task design,

2. A larger-scale evaluation of the Random Prior (RP) (Ciosek et al. 2019) and RotNet (Hendrycks, Mazeika, and Dietterich 2019) methods, and

3. A novel OOD detection method.

Chapter 2 places OOD detection in the context of anomaly detection, uncertainty and robustness, defines reference measures for OOD detection, and reviews previous methods and evaluations. In Chapter 3, we explain the benchmark design. Chapter 4 investigates a neural network architecture called TwinNet and its use in the novel method. Benchmark results are in Chapter 5, and Chapter 6 concludes.

Chapter 2

Background and related work

2.1 OOD detection in context

Out-of-Distribution (OOD) detection has several closely related problems. In this section, we situate OOD detection in the context of these problems using qualitative distinctions. For a precise definition of OOD from a design perspective, refer to Definition 1, and for the OOD detection task, refer to Section 3.1.

2.1.1 Anomaly detection

In a technical sense, OOD detection problems are *anomaly detection* problems. An anomaly detector is given a set of data points and assigns an anomaly score to each, with higher score indicating stronger belief that the point is anomalous (Emmott et al. 2016). Anomaly detection is often synonomous with *outlier detection*, where anomalies are assumed to be statistical outliers (Emmott et al. 2016). However, OOD detection is more specific: it assumes that in addition to the data, we have a pretrained predictive model. In the scope of our benchmark, this model classifies images. While the model does not add useful information for the task[1], it provides potentially useful representations of the data distribution. These representations may improve the efficiency of detection over unsupervised alternatives, though we believe this is an open question.

We are more interested in how the predictive model handles anomalies than the nature of the anomalies. Many other works seem to have a similar motivation (Hendrycks and Gimpel 2016; Liang, Li, and Srikant 2018; Lee et al. 2018a; Shafaei 2019; Meinke

[1]Information may be added in the form of inductive bias, but in principle the inductive bias could be incorporated into the anomaly detector.

and Hein 2020), with notable exceptions (Ahmed and Courville 2019). *Novelty detection* also tends have a different motivation, such as exploration in Reinforcement Learning (Burda et al. 2018).

2.1.2 Uncertainty and robustness in deep learning

To our knowledge, almost all work that is explicitly about OOD detection deals with Deep Neural Networks (DNNs) as the supervised models and, often, as part of the OOD detector. It is well known that performance of baseline DNNs is easily compromised by data shift (Rabanser, Günnemann, and Lipton 2019). In addition, uncertainty estimates and likelihoods of baseline DNNs are inaccurate on OOD data (Guo et al. 2017; Nalisnick et al. 2019). When DNNs "don't know what they don't know" they can fail silently, posing safety and security risks in real-world applications. Uncertainty estimation and confidence calibration are thus strongly related to OOD detection. In fact, uncertainty estimates often double as OOD detection scores and vice versa (Lee et al. 2018a; Hendrycks, Mazeika, and Dietterich 2019; Meinke and Hein 2020).

2.2 Defining out-of-distribution

A benchmark designer defines OOD in practice by specifying datasets. We assume that the benchmark designer has training data generated from an *in-distribution* p_i—where "training data" includes the test set used for evaluation—and OOD data generated from an *out-distribution* p_w. Definition 1[2] then defines a reference score and category for OOD.

Definition 1. Out-of-distribution reference score, and reference categorisation.

Suppose we are given a distribution $p_i(x)$ that generates the training data, a distribution $p_w(x)$ that generates data from a wide space of possible datasets (including the training data), and a threshold $\gamma \in \mathbb{R}$.

Then given a query point x', we define an out-of-distribution reference score as

$$s = log\frac{p_w(x')}{p_i(x')} = \log p_w(x') - \log p_i(x') \tag{2.1}$$

Likewise we can apply a reference category: x is out-of-distribution if $s > \gamma$.

[2]The definition was provided by the supervisor.

Note how the definition does not exclude the training data from p_w. This is deliberate, as it allows the evaluation of different datasets against the same broader "world model", and enables consistency. It is straightfoward to compute a recalibrated score against p_w not including p_i. It is also important to realise that the distributions p_i and p_w are generally unknown in practice, and benchmark users must estimate them from data (implicitly or explicitly). In turn, the reference score s may not be tractable to compute, hence the need for proxy scores (see Section 2.3.2). Furthermore, the precise nature of OOD depends on the value of γ. In practice the detection threshold should be set according to the data and risk involved in the application domain. The metrics of our benchmark (Section 3.5) are agnostic to threshold.

2.3 Approaches to OOD detection

2.3.1 Task and sources of data

We have identified two main types of OOD detection task in the literature: *dataset-based* and *class-based*. As a concrete example throughout this section, we consider training a Deep Neural Network (DNN) classifier on CIFAR-10.

A dataset-based task treats some distinct, irrelevant dataset as OOD. For example, LSUN (Netzer et al. 2011) consists of images of scenes, such as classrooms and bridges, which do not relate to any CIFAR-10 class. At minimum, such OOD data must have the same input format as the in-distribution, e.g. a 3-channel image. Beyond that, choices of OOD dataset vary in similarity to the in-distribution, both in low- and high-level information. The dataset may have no overlapping classes and come from quite a different input distribution, such as LSUN (Liang, Li, and Srikant 2018; Hendrycks, Mazeika, and Dietterich 2019). Alternatively, no classes overlap but the input distribution is quite similar, such as CIFAR-100 (Hendrycks, Mazeika, and Dietterich 2019). Another choice of OOD data comes from a distinct dataset where some or all classes are similar, e.g. Tiny ImageNet (Liang, Li, and Srikant 2018; Shafaei 2019). We prioritise the unknown-class case because it is not directly solved by increased robustness, and therefore seems to be more neglected.

In dataset-based tasks one is free to select datasets for certain characteristics, such as how similar they appear to the training data. Based on the OOD reference score (Equation 2.1) we would expect less similar data to be easier to detect. However, neural networks can be biased in a way that makes them particularly vulnerable far away from

the training data (Hein, Andriushchenko, and Bitterwolf 2019). A prime example of a very dissimilar dataset is random noise. Several distributions have been used to generate noise OOD data, including uniform, Gaussian, and Bernoulli (Liang, Li, and Srikant 2018; Hendrycks, Mazeika, and Dietterich 2019). Other kinds of far away data include blobs and textures (Hendrycks, Mazeika, and Dietterich 2019). Though unrealistic, these datasets provide useful diagnostics for research. For example, a large difference in ability to detect Gaussian versus Bernoulli noise indicates a systematic bias that may warrant further investigation.

The second type of OOD detection task is *class-based*. **A class-based task partitions the classes of some dataset in two, and treats one partition as OOD**. For example, training on data with the labels {dog, horse, ship, automobile} and evaluating detection of {cat, deer, airplane, truck} examples in the test set. There are two important comparisons to make with dataset-based tasks. First, the covariate distributions are much more similar than is typical for dataset-based tasks, because they come from the same original dataset. Second, there is a stronger conceptual argument that the excluded classes have no valid label; they are *semantic* anomalies (Ahmed and Courville 2019). This is because all classes were clearly defined and mutually exclusive in the original dataset. The use of class-based tasks varies in how many classes are excluded, ranging from one class (Ahmed and Courville 2019), to several (Ciosek et al. 2019), to all-but-one (Schölkopf et al. 2000; Hendrycks, Mazeika, Kadavath, et al. 2019).

Note that class-based tasks are not completely distinct from dataset-based tasks. For example, CIFAR-10 and CIFAR-100 are separate datasets, but the images of both come from the 80 Million Tiny Images dataset (Torralba, Fergus, and Freeman 2008). However, since the datasets were curated and labelled separately (Krizhevsky 2009), we consider detecting CIFAR-100 with a CIFAR-10 model to be a dataset-based task.

A prime example of the class-based task is Ahmed and Courville 2019, because they argue extensively for it as an alternative to typical dataset-based tasks. Their benchmark excludes each individual class of a dataset in turn, taking the average as the overall metric. We believe this is a more informative and well-motivated benchmark than most previous work. However, it has at least two limitations. First, excluding each class individually requires training a number of models in proportion to the number of classes in the dataset. This is somewhat tractable for 6-12 classes, as in Ahmed and Courville 2019, but it does not scale well to 100s or 1000s of classes, e.g. CIFAR-100 or ImageNet respectively. So while the datasets used by Ahmed and Courville 2019 serve well for standard benchmarking, there is still reason to benchmark using e.g. ImageNet

if it is more applicable.

The second limitation of the benchmark in Ahmed and Courville 2019 is that excluded class anomalies are not the only practically important kind, and in general, anomalies do not belong to one semantic category. Aggregating over the classes indicates how well a detection method generalises, but does not substitute for detecting multiple classes at once. As we show in Section 3.2.1, the argument that previous dataset-based benchmarks are trivial is not well-supported. However, the responsibility for this lies with Liang, Li, and Srikant 2018 and the way they pre-processed OOD data, not with Ahmed and Courville 2019.

2.3.2 Score

The detection scores used by practical methods are effectively proxy measures for the reference score in Equation 2.1. One approach to scoring uses likelihood: strictly, the negative log likelihood of the query data point $\mathbf{x}'$ under a generative model. There are many approaches to learning the generative model from the training data, ranging from classic methods common in the anomaly detection literature, such as k nearest-neighbours (Shafaei 2019), to deep generative models such as PixelCNN (Hendrycks, Mazeika, and Dietterich 2019).

Another type of detection score is uncertainty. While likelihood is associated with inputs, uncertainty is associated with predictions. We further distinguish *first-order* and *second-order* uncertainty: first-order uncertainty relates to predictive confidence, e.g. the output probability of a classifier, while second-order uncertainty measures the uncertainty in the confidence itself. The first use of classifier probability in an OOD detection score is Hendrycks and Gimpel 2016, with the negative Maximum Softmax Probability (MSP). Later first-order methods still use MSP, but add various techniques during training to improve score quality. Deep Ensemble (DE) (Lakshminarayanan, Pritzel, and Blundell 2017) employs adversarial training of multiple independent copies of the model, then uses the average over their predictive probabilities. Meanwhile, ODIN (Liang, Li, and Srikant 2018) uses temperature tuning and adversarial perturbations at *inference* to achieve similar benefits. "GAN" (Lee et al. 2018b) augments the loss function to enforce low confidence on synthetic OOD examples generated by a Generative Adversarial Networks (GAN) model. Outlier Exposure (OE) (Hendrycks, Mazeika, and Dietterich 2019) similarly augments the loss function, but uses a large set of *natural* OOD examples. Certified Certain Uncertainty (CCU) (Meinke and

Hein 2020) extends this by estimating the density of inputs under the in- and out-distribution separately. This affords provable guarantees on the confidence of the classifier, while maintaining competitive detection quality. Finally RotNet (Hendrycks, Mazeika, Kadavath, et al. 2019; Ahmed and Courville 2019) uses auxilliary tasks such as rotation prediction during training. This improves detection of OOD data supposedly by producing better feature representations.

An advantage of second-order uncertainty is the decoupling of prediction uncertainty from model uncertainty (Leibig et al. 2017). The second-order methods we are aware of use a variance estimate. Monte Carlo Dropout (MCD) (Gal and Ghahramani 2016) leaves dropout turned on during inference to make predictions stochastic. For a given input, uncertainty is then estimated by the standard deviation of a sample of predictions. However, the mean prediction of MCD has also been used as a baseline score method in Ciosek et al. (2019). The Random Prior (RP) method proposed by Ciosek et al. (2019) is very different to MCD, as it distils a random network into a trained one and uses the Mean Squared Error (MSE) between them as the uncertainty.

2.3.3 Evaluation

Benchmarks are the measure and driver of progress in empirical machine learning. The quality of a benchmark is therefore crucial. Unfortunately there is so far no consensus on evaluating OOD detection. However, there are multiple benchmarks aiming for wider adoption (Shafaei 2019; Ahmed and Courville 2019; Hendrycks, Zhao, et al. 2020). One objective of this thesis is to learn from and critique past benchmarks, in order to develop an even better one. In what follows, we outline the key issues with evaluation and how to address them.

One key issue with evaluation is the tuning procedure. Many OOD detection methods require tuning the hyperparameters of the detector on some validation dataset in order to perform well on OOD test samples. There are several strategies for this, some of which we find problematic. For instance, ODIN (Liang, Li, and Srikant 2018) requires tuning temperature and adversarial perturbation parameters. For the validation dataset, both Liang, Li, and Srikant 2018 and Lee et al. 2018b opt for a disjoint subset of the same dataset used for testing. This is problematic because the evaluation then fails to indicate how the model generalises to *unseen* OOD distributions. This ability is crucial, because by the nature of OOD, we generally cannot anticipate all sources of OOD data. With that said, less biased evaluations of ODIN have since been performed (Shafaei

2019; Ahmed and Courville 2019) which confirm its strong performance compared to the MSP baseline.

An alternative to using the same distribution for validation is to use a representative but different OOD distribution. Meinke and Hein 2020 opt for a large set of natural images that is disjoint from the test datasets. However, this still assumes some prior knowledge of the out-distribution, and access to such a large separate dataset. Prior knowledge may not be problematic, and should be exploited in real-world applications. However, we argue that it should be minimised in a benchmark for basic research, to make it generic and widely applicable. Meanwhile, Shafaei 2019 employ a cross-validation style of benchmark, where each dataset in a collection is successively held out for validation, the remaining datasets are used for testing, and the results are averaged. This has the advantage of being agnostic, in aggregate, about which dataset is used for validation. However, the individual evaluation runs still rely on a specific validation set, with ideosyncrasies that do not necessarily cancel out in aggregate. The chosen collection of datasets therefore might still be biased. Indeed, when running this benchmark using real medical datasets rather than standard natural image datasets, Cao et al. 2020 find the rank order of methods is roughly the reverse of that in Shafaei 2019. In fairness, the same issue could manifest in other benchmarks, but this is the only case we are aware of.

A third approach to validation, which we adopt for this thesis, is to perform various drastic transformations of the training data to render it OOD. This is used by Hendrycks, Mazeika, and Dietterich 2019, where the transformations include severe pixellation, jigsaw-like permutations of image patches, and colour inversion. This approach is flexible yet does not require assumptions about what kind of OOD data will be seen.

Another issue with previous evaluations is a lack of diversity in datasets. Diversity is important because the out-distribution is so broad, and performance can be very sensitive to the dataset. For example, Ciosek et al. (2019) and Hendrycks, Mazeika, Kadavath, et al. (2019) only use CIFAR-10 as an in-distribution for dataset-based tasks. Ciosek et al. (2019) is also short on OOD datasets, only using a subset of CIFAR-10 and SVHN. Good examples of diversity in datasets are Hendrycks, Mazeika, and Dietterich (2019) and Meinke and Hein (2020), which evaluate five training datasets with at least five OOD datasets for each.

Many evaluations of OOD detection also lack baselines. For early methods such as Hendrycks and Gimpel (2016) and Liang, Li, and Srikant (2018) this is more acceptable, although they could still have used unsupervised baselines from the anomaly detection

domain. Even if the objective is specifically to make neural networks good at OOD detection, comparing to other types of method indicates marginal costs. For example, if more computation is required during inference to calibrate a network, a separate fixed model might be faster. Some more recent works still lack baselines—for example, Hendrycks, Mazeika, Kadavath, et al. (2019) only compares to MSP for the dataset-based task, when methods such as ODIN, GAN, and Outlier Exposure were available.

Finally, we believe that most previous work considers too narrow a set of evaluation metrics. To evaluate detection, Shafaei (2019) only consider accuracy at one detection threshold per method and dataset, while Ahmed and Courville (2019) only consider Average Precision (AP). Hendrycks, Mazeika, and Dietterich (2019) use a broader set, reporting Area Under the Receiver Operating Characteristic (AUROC), AP and False Positive Rate at true positive rate 95% (FPR95). Different detection metrics will vary in importance for different users, so reporting multiple metrics will satisfy more users and indicate trade-offs. However, we believe that a complete evaluation of OOD detection also includes metrics that are not directly related to detection. If the method modifies the baseline classifier in some way, then it is important to check the effect on test error and calibration. Furthermore, efficiency is a potentially critical yet neglected aspect of evaluation. If a method is too inefficient then it may be unsuitable, no matter if OOD detection is perfect. It is therefore important to report the inference time, ideally separating OOD score time from the classifier prediction time to indicate the marginal cost of detection. Besides the qualitative assessment in Shafaei (2019), we are not aware of work that reports efficiency metrics for OOD detection.

Chapter 3

Benchmark design

3.1 Task definition

Benchmark users are given a collection of training datasets $\mathcal{D}_{\text{tr}}$. Each dataset consists of pairs of inputs $x \in \mathcal{X}$ and their labels $y \in \mathcal{Y}$. Benchmark users are also given a pretrained, supervised model $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ for each training dataset. Based on the training data and pretrained models, users must build an Out-of-Distribution (OOD) detector. In evaluation, the detector observes query points x' one at a time. The query points are drawn at random between an in-distribution test set D_i^{test} and an OOD test set $\mathcal{D}_o^{\text{test}}$, with their union forming $\mathcal{D}^{\text{test}}$. The OOD detection task is defined as $\mathcal{T} : \mathcal{D}^{\text{test}} \to \{-1, 1\}$, that is, predict whether $x' \in \mathcal{D}_o^{\text{test}}$. We adopt the convention of $\mathcal{D}_o^{\text{test}}$ being the positive class (1) and $\mathcal{D}_i^{\text{test}}$ being the negative class (-1). However, **the task for the user's detector is to output a score** $s \in \mathbb{R}$ **rather than the class**. Note that the score is more general and includes the binary class values.

3.2 Preliminary experiments that informed the design

3.2.1 Performance can be very sensitive to preprocessing

Simple baselines are a low-cost way to evaluate a benchmark. In particular, if a simple method performs very well, it suggests that the benchmark is too easy. A good benchmark should be difficult even for state-of-the-art methods. This is evidence that the benchmark meets its purpose as a genuine research challenge and a longstanding, meaningful measure of progress.

To help evaluate the quality of dataset-based benchmarks, we followed Ahmed and

Courville 2019 by examining the performance of a Gaussian mixture model (GMM). The GMM we used was especially simple, following the description in Appendix D of Ahmed and Courville 2019. It fits one multivariate Gaussian distribution to each image channel independently. The mean and covariance parameters were computed directly as the maximum likelihood estimates, with full covariance, namely

$$\mathbf{m}_c = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_{c,i} \quad , \quad S_c = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_{c,i}-\mathbf{m}_c)(\mathbf{x}_{c,i}-\mathbf{m}_c)^{\mathrm{T}} \tag{3.1}$$

where c is the image channel index, $\mathbf{x}_{c,i}$ is the flattened vector form of channel array c, and i indicates the index in a dataset of N samples.

We fixed the mixture weights of the channel-wise components to be equal, with no optimisation. It is unclear whether Ahmed and Courville 2019 optimised the weights. However, preliminary experiments with CIFAR-10 as the in-distribution and Tiny ImageNet as the out-distribution showed that weight optimisation had a negligible effect on OOD detection performance.

We initially replicated the results in Appendix D of Ahmed and Courville 2019, training on CIFAR-10 and testing on Tiny ImageNet as OOD. This required transforming Tiny ImageNet samples from size 64x64 to 32x32, both via cropping and resizing. We noticed that the results for resizing only replicated under nearest-neighbour resampling, and varied greatly under different resampling. This led to the results in Table 3.1. It lists average precision scores for OOD detection using the GMM. These scores are in turn an average over the following OOD datasets (excluding whichever is used for training): SVHN, CIFAR-10, CIFAR-100, STL-10, LSUN, Tiny ImageNet, Gaussian noise, Rademacher noise, and the describable textures dataset (Cimpoi et al. 2014).

We see that the performance of the GMM can be very sensitive to different image resampling methods. This is true for the CIFAR datasets in particular: the average precision ranges from roughly 40% under bilinear interpolation to 77% under nearest-neighbour interpolation. Note that CIFAR images were most likely downscaled via bicubic interpolation originally (we infer this from information in Torralba, Fergus, and Freeman 2008, but it is not certain). Meanwhile, Tiny ImageNet was downscaled using bilinear interpolation[1] . We suspect that bilinear interpolation is even more difficult than bicubic for the CIFAR datasets because of the phenomenon described in Nalisnick et al. 2019, where the out-distribution lies "inside" the in-distribution, but we did not investigate further.

[1] See `https://github.com/jcjohnson/tiny-imagenet/`

Table 3.1: Average precision of OOD detection for a channel-wise Gaussian mixture model. The columns denote the method of image interpolation, where applicable.

In-distribution	BICUBIC	BILINEAR	BOX	NEAREST
SVHN	98.5	97.8	98.9	99.3
CIFAR10	53.2	40.1	63.8	76.6
CIFAR100	52.5	40.2	63.3	76.7
TinyImageNet	23.5	23.2	23.6	27.6

The results in Table 3.1 had two implications for our benchmark design. The first is how the benchmark handles different image sizes between the in- and out-distribution. If the in-distribution images are resampled from some original source, then the out-distribution images are resampled in the same way. Otherwise, if the in-distribution data was not resampled in the first place, or there is no evidence of how it was resampled, the default was bilinear interpolation[2]. In general, the policy for preprocessing images was to be consistent between in- and out-distribution. This is realistic, because preprocessing must occur before OOD detection, so it cannot discriminate between the distributions. Using the same preprocessing also tends to make the distributions more similar than otherwise, making detection more challenging.

The second implication of the results in Table 3.1 is that detecting OOD datasets relative to SVHN is almost solved. In the worst case of bilinear interpolation, the simple GMM baseline achieves 97.8% average precision. As such, we do not use SVHN as an in-distribution in subsequent experiments.

Finally, we want to emphasise how this result weakens the evidence for the claim in Ahmed and Courville 2019 that dataset-based benchmarks used in earlier work are trivial. By using consistent and appropriate resampling, we show that this GMM baseline is far from solving the task in the case of the CIFAR and ImageNet datasets. There is a clear need for alternative methods to close the gap. We therefore use this dataset-based task as part of our benchmark.

[2]Bilinear interpolation is also the default for `torchvision.transforms.Resize` in the PyTorch library.

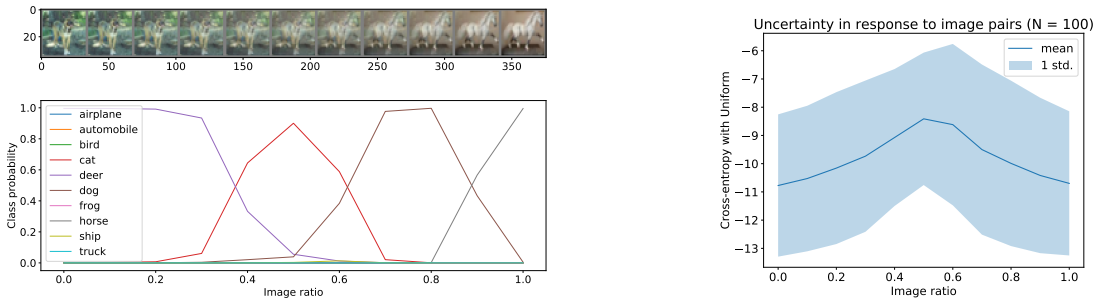

Figure 3.1: The probability distribution of a CIFAR-10 classifier over an averaged pair of images often involves overconfidence in seemingly arbitrary classes (left). In aggregate the model is more uncertain about mixtures, but with high variance (right).

3.2.2 Response to corrupted training data

Towards designing the validation task, Figure 3.1 shows one example and an aggregation of a model's response to averaged pairs of images. While the model is appropriately uncertain in broad aggregate, it is highly variable and often overconfident. This indicates that averaged images, though contrived, are relevant to the problem we are aiming to solve, providing a useful indication of performance for validating methods. We use averaged image pairs and other operations for the validation set in this thesis; see Section 3.3.2.

3.3 Data sources and preprocessing

Datasets are a crucial component of benchmark design. While we have already established the task of OOD detection in abstract, the datasets are what characterise the task in practice. In the following sections we explain the content, selection and organisation of datasets for our benchmark. Section 3.3.1 covers the in-distribution, Section 3.3.2 covers OOD for validation, and Section 3.3.3 covers OOD for evaluation. Some statistics for the off-the-shelf datasets that we used, or their derivatives, are listed in Table 3.2.

3.3.1 In-distribution

For in-distribution datasets, we chose CIFAR-10, CIFAR-100, and Tiny ImageNet. All of these datasets consist of natural colour images of common objects—the class names are listed in Appendix D.1. The datasets are common in the computer vision and OOD detection literature already (Hendrycks, Mazeika, and Dietterich 2019; Shafaei 2019;

Name	Train size	Test size	Dimensions	Classes
SVHN	73,257	26,032	3072	10
CIFAR-10	50,000	10,000	3072	10
CIFAR-100	50,000	10,000	3072	100
Tiny ImageNet	100,000	10,000	12,288	200
ImageNet-x	-	40,000	12,288	800
ImageNet-O	-	2000	547,698	200
LSUN	-	10,000	94,396*	10
Textures	-	5640	689,850*	47

Table 3.2: Statistics of off-the-shelf datasets (or their derivatives) used to evaluate OOD detection. *Dimensions vary so we list the mean.

Meinke and Hein 2020), which affords greater accessibility and closer comparison to previous work. The small dimension of these datasets slightly limits applicability, since many images of real-world relevance are much larger (e.g. those found in ImageNet-O, see Table 3.2). However, the large difference in dimension between CIFAR datasets and Tiny ImageNet provides evidence of how methods scale. Furthermore, smaller dimension demands less computational resources and in turn makes the benchmark more accessible.

Another reason we chose these datasets is for the variety in number of classes: 10, 100 and 200 respectively. This allows us to evaluate how methods scale with output dimension as well as input dimension. Note that we do not use MNIST or its derivatives e.g. FashionMNIST (Xiao, Rasul, and Vollgraf 2017). While these are commonly used, the input data is grayscale and very sparse compared to the chosen datasets. It is well recognised in the machine learning field that MNIST datasets are relatively uninformative, because they are too easy and the results often do not generalise well to other domains. We expected CIFAR-10 to be more informative as a dataset with low input and output dimension.

3.3.2 OOD validation

We use a validation dataset to measure the performance of OOD detection methods before the final evaluation. This is useful for tuning hyperparameters and guiding the development of new methods, without compromising the fairness of evaluation. For the supervised model $\mathcal{M}$, it is acceptable to draw the validation dataset from the same

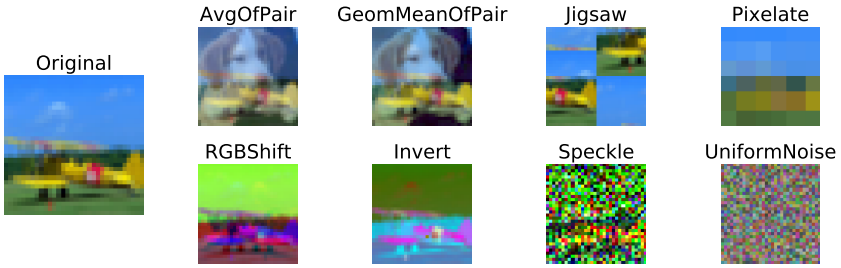

Figure 3.2: Examples of each type of validation data

distribution as the evaluation dataset, as long as the particular datasets are disjoint. However, the out-distribution is made up of many natural "kinds", e.g. Gaussian noise, or images of an unknown animal. Generalisation to unseen kinds is central to the task, because OOD examples are unexpected and unfamiliar by nature. The validation dataset should therefore be sampled from a different *distribution* to evaluation, ideally a disjoint one. However, this constraint makes it more difficult to get information about performance that correlates with the final evaluation.

Our validation dataset was constructed by applying various tranformation and mixture operations on the training dataset. These operations were sourced from Hendrycks, Mazeika, and Dietterich 2019, namely: arithmetic and geometric means of randomised pairs of images, a fixed permutation of rectangular regions ("jigsaw"), severe pixellation, colour offset, inversion, severe speckle noise, and uniform noise (the only one which does not use the training dataset). Each operation is illustrated for one example image in Fig. 3.2. The dataset was not intended to be realistic, but merely representative of OOD.

We emphasise that the validation dataset described here was chosen merely for this thesis, and is not part of the benchmark design. The benchmark user is free to use any validation dataset of their choice under one important condition: it must be derived either from generic operations on the training data, or from generic noise distributions that are not used in evaluation, such as uniform noise. The term "generic" is meant to avoid perverse operations, such as adding pixel values to transform training data into known OOD data. We set this condition for two reasons. Firstly, it results in minimal assumptions about the OOD data observed in deployment, ensuring that new examples are unfamiliar. In practical applications, there probably *will* be useful prior knowledge about the kinds of OOD data that will be encountered. However, for a generic benchmark such as ours, it is best to be agnostic. Secondly, the condition makes validation data more resource-efficient, in the sense that it does not use an extra OOD

dataset, and that unlimited amounts of OOD data can be constructed by corrupting the training data or generating noise.

3.3.3 OOD evaluation

The results of the GMM experiment in Section 3.2.1 indicated that dataset-based benchmarks are far from trivial, contrary to Ahmed and Courville (2019). However, class-based benchmarks are still appealing in that the difference between in- and out-distribution tends to be more semantic and clearly defined, and the task tends to be more challenging (to see this, compare average precision between tasks in Hendrycks, Mazeika, and Dietterich (2019) and Ahmed and Courville (2019)). Given the merits of both, our benchmark includes a dataset-based and class-based task.

For the dataset-based task, we aimed for a relatively large and diverse collection of OOD datasets to make results more generalisable. The collection is similar to Hendrycks, Mazeika, and Dietterich 2019. SVHN (Netzer et al. 2011) consists of close-up images of house numbers. LSUN (Yu et al. 2016) consists of scenes such as *classroom* and *bridge*. Textures (Cimpoi et al. 2014) consists of both real and synthetic images that are naturally described by one texture, e.g. *bubbly* and *wrinkled*. We include Textures because convolutional networks tend to rely on textures as cues, and Hendrycks, Mazeika, and Dietterich (2019) found it was relatively difficult to detect using the Maximum Softmax Probability (MSP) score. Finally, "Gaussian" is Gaussian noise and "Rademacher" is Rademacher noise (i.e. each dimension is selected uniformly at random from $\{-1, 1\}$). We chose these noise datasets for variety and because in some cases even the very effective Outlier Exposure method is poor (see Hendrycks, Mazeika, and Dietterich (2019) Table 7). The noise datasets are independently generated for each evaluation run in the same size as the in-distribution data.

For all three in-distribution datasets, the out-distribution datasets include Gaussian, Rademacher, Textures, SVHN, and LSUN. Then, for CIFAR-10 and CIFAR-100, we include CIFAR-100 and CIFAR-10 respectively, expecting this to be particularly difficult due to similarity. Note that CIFAR-10 and CIFAR-100 classes are disjunct. For the same reason, we use ImageNet-x for Tiny ImageNet. ImageNet-x is the ILSVRC 2012 validation set (Russakovsky et al. 2015) minus the subset used for Tiny ImageNet, and downscaled in the same manner as Tiny ImageNet. We also include ImageNet-O (Hendrycks, Zhao, et al. 2020), a set of difficult OOD examples for ImageNet classifiers. ImageNet-O samples are sourced from the 22,000-class version of ImageNet and are

disjoint from the 1000-class version. All together, each in-distribution has 6 or 7 dataset-based subtasks, with two based on noise, one on textures, two with perceptably different semantics and low-level features, and at least one with very similar low-level features but different semantics.

For the class-based task, we construct a fold (i.e. subset of classes) by pseudo-randomly choosing half of the total classes. We use 10 of these folds for each in-distribution dataset. Random selection reduces bias, and partitioning in half allows more possible combinations of classes while reducing the number of folds necessary. The latter is a limitation of one-class (Schölkopf et al. 2000; Hendrycks, Mazeika, Kadavath, et al. 2019) and all-but-one class (Ahmed and Courville 2019) tasks, which require a number of folds equal to the number of classes. A downside of our approach is that the random subsets lack a natural interpretation. Training 10 models per training dataset is also burdensome for the user, but worthwhile to give reliable results.

3.3.4 Preprocessing

Since OOD detection is unsupervised, our benchmark assumes that data preprocessing is consistent between in- and out-distributions. The preliminary GMM experiment (Section 3.2.1) also indicated that using consistent preprocessing between the in- and out-distribution makes detection more difficult. To reiterate, greater difficulty is good at the margin because it makes the benchmark a longer-lasting research target. Given the evidence that performance is highly sensitive to the method of resampling, we resample OOD data consistently using bicubic interpolation for CIFAR-10 and CIFAR-100, and bilinear interpolation for Tiny ImageNet.

3.4 Baselines

A baseline method provides a point of comparison for a new method. Having at least one baseline is essential, but using multiple baselines affords a nuanced perspective on the strengths and limitations of any new method. For example, one baseline excels in accuracy, another in efficiency, and a new method is intermediate on both metrics—the best of these three will depend on the application domain. We now briefly explain and justify the baseline methods chosen for evaluation. We prioritise methods that are recent, controversial (past evaluations conflict), or neglected (lacking large-scale evaluation). Besides the methods listed below, there is the GMM which was explained

in Section 3.2.1, and our proposed TwinNet ensemble method which is explained in Chapter 4. Comparing to the GMM indicates the marginal value of using Deep Neural Networks (DNNs), including but not limited to the supervised model, $\mathcal{M}$.

For all instances of $\mathcal{M}$ we use a 40-2 Wide ResNet (Zagoruyko and Komodakis 2017). ResNet architectures are popular for computer vision, and this one provides a good trade-off between efficiency and accuracy. The same arcthiecture is used in previous benchmarks OOD detection (Hendrycks, Mazeika, and Dietterich 2019; Ahmed and Courville 2019). Further details on the model are in Appendix B.1.

3.4.1 Plain network (MSP)

We expect a well-calibrated probabilistic classifier to have lower confidence on less familiar inputs. The plain MSP baseline works under this assumption, using the confidence of $\mathcal{M}$ to detect OOD examples. It was first used by Hendrycks and Gimpel 2016. At its final layer, $\mathcal{M}$ outputs a vector of logits l, of length K. These are then passed through a softmax operation to produce a categorical probability distribution over the class $k \in \{0 \dots K-1\}$: $\hat{p}(y=k) = \exp(l_k) / \sum_{j=0}^{K-1} \exp(l_j)$. The prediction $\hat{y}$ is then given by $\arg\max_k \hat{p}(y=k)$, with an associated confidence of $p(y=\hat{y}) = \max_k \hat{p}(y=k)$. The MSP detection score is then $-p(y=\hat{y})$, with the negation making higher scores correspond to greater belief in OOD.

3.4.2 Random Prior (RP)

Random Prior (RP) employs a pair of neural networks, which may be repeated in an ensemble Ciosek et al. 2019. The *prior* network is randomly initialised with fixed parameters. A scalar hyperparameter c controls the scale of the prior parameters. Meanwhile, the *learner* has a similar initialisation and architecture to the prior, but with some extra layers appended in order to make it more expressive than the prior. Note that both of these networks are distinct from $\mathcal{M}$, though they conventionally have similar architecture. The prior and learner both output an arbitrary vector of the same length. The task of the learner is to predict the output of the prior, training on the same dataset as $\mathcal{M}$. After training, the Mean Squared Error (MSE) between the learner and prior on new inputs is used as an uncertainty estimate, which also serves as an OOD detection score (Ciosek et al. 2019). An intuition for why RP works is that by fitting a *random* prior, which is barely informative of the task, the learner becomes exceedingly overfitted to the training data. Any OOD data it encounters should then

give an exceedingly different output to the prior, making the MSE effective as an OOD score. A simple example of the phenomenon is illustrated in Fig. A.2

3.4.3 Monte Carlo Dropout (MCD)

Dropout Srivastava et al. 2014 is a module for neural networks that randomly sets input activations to 0 with probability p. Conventionally, this module is only applied during training to reduce co-adaptation of network units and in turn reduce overfitting. For inference, conventional dropout simply multiplies the activations by p to approximate the effect of averaging predictions from different dropout settings. In contrast, Monte Carlo Dropout (MCD) (Gal and Ghahramani 2016) keeps dropout active for inference and explicitly samples the network multiples times. Using the mean predictive probability from these samples has been shown to improve test error in numerous domains (Gal and Ghahramani 2016). Meanwhile, the standard deviation of the predictive probability can provide useful uncertainty estimates (Leibig et al. 2017). However, based on previous work it is unclear how useful MCD is for OOD detection. On one hand, Meinke and Hein (2020) find that MCD performs much worse than a plain neural network. On the other hand, Shafaei (2019) find it is slightly better. Given the inconsistent results, we include MCD to provide more evidence. It is also a natural baseline for the novel TwinNet method, because both methods aggregate over randomised subnetworks (see Chapter 4). We use the MSP score of the mean predictive probability rather than the standard deviation as it performed better on the validation set.

3.4.4 RotNet

Our RotNet baseline followed the method of Hendrycks, Mazeika, Kadavath, et al. (2019), which in turn is based on Komodakis (2018). For RotNet one trains an additional linear classifier ("head") on top of the penultimate feature layer of $\mathcal{M}$. The task of the additional classifier is to predict rotations of the input image, outputting $p_{\text{rot\_head}}(r \mid R_r(x))$ where R_r is a rotation transformation by angle r. In turn, RotNet must be fed different rotations $R_r(x)$ as input. We used the standard $r \in \{0°, 90°, 180°, 270°\}$ for each sample in each batch. The training loss was the same as in Hendrycks, Mazeika, Kadavath, et al. (2019) with $\lambda = 0.5$, along with the detection score

$$s = -\mathrm{KL}\left[U \,||\, p(y|x)\right] + \frac{1}{4} \sum_{r \in \{0°, 90°, 180°, 270°\}} \mathcal{L}_{\mathrm{CE}}\left[\text{one\_hot}(r), p_{\text{rot\_head}}(r \mid R_r(x))\right] \quad (3.2)$$

where U denotes a uniform distribution over y. The KL divergence is used in the first term to be more compatible with the second term, as in this case the cross-entropy is equal to KL divergence. Note that by construction, we have labels for the rotation task even during inference, whereas the original classifier relies on confidence alone.

3.5 Metrics

We aimed for a comprehensive set of metrics to give a full profile of each method. First, there are metrics related to the base task. The most important of these is the classification error on $\mathcal{D}_{\text{te,IID}}$, being the fraction of samples that were incorrectly classified. The benchmark also assesses calibration using the Root Mean Square (RMS) calibration error. Finally, the benchmark includes metrics for error detection, the task of classifying whether $\mathcal{M}$ predicted incorrectly, based on the MSP score. The detection metrics were the same as for OOD detection, explained next.

To evaluate OOD detection, the benchmark uses three evaluation metrics. The first two, Area Under the Receiver Operating Characteristic (AUROC) and Average Precision (AP), aggregate detection performance over all possible thresholds on the set of scores. Higher values of AUROC and AP indicate better overall detection. AUROC tends to be more optimistic when positives are rare (Davis and Goadrich 2006), with a random-chance level of 50%, while the AP random-chance level is the relative frequency of the positive class. This is also known as *skew*, which we fixed at 16.67% for all dataset-based experiments and 50% for all class-based experiments. The third metric is the false positive rate when the true positive rate is fixed at 95% (FPR95). This indicates the cost of false alarms when almost all positives are recalled, meaning that lower FPR95 is better.

Finally, our benchmark evaluates the efficiency of methods by time metrics. For training, it measures the time to iterate the training set and test set at each epoch. Note that these times include operations that are not specific to the method, such as computing accuracy, but this was kept consistent for all experiments. For inference time we measure the average time to execute the supervised model (if used) or on one batch, while score time is the time to compute the OOD detection score on an entire dataset. We average these over the in- and out-distribution datasets for the dataset-based task.

3.6 Reproducibility and statistical significance

For reproducibility, each training and evaluation run was completely deterministic for a given random seed. This allowed methods that built upon a pretrained model from some previous experiment (e.g. TwinNet) to automatically use the same class subsets. Furthermore, each experiment run had a date-time stamp as an interpretable ID, and experimental settings were saved to disk. We also partitioned the evaluation pipeline into raw outputs, detection scores and metrics, to make the benchmark fault-tolerant.

Considering statistical significance, we deemed the cost of multiple training runs too prohibitive. As an alternative we evaluate each method for 10 trials. Each trial uses a randomised, fixed-size subset of scores on each OOD dataset. Furthermore, though we did not repeat any one training setting, we did repeat the class-based task for 10 independent runs. This produced 10 models per in-distribution dataset, each trained on a different subset of classes. We averaged over these class subsets to summarise the performance of each method.

3.7 Benchmark design specification

Our benchmark works as follows. Users are given the training datasets (see Section 3.3.1) and the pretrained classifier $\mathcal{M}$ for each. The OOD validation dataset used in this thesis is also provided, but users are free to design their own subject to constraints (see Section 3.3.2). In turn, the user provides a detector model for each training dataset. Given an image, the detector must output a single real-valued score which is proportional to the belief that the image is OOD. The evaluation program runs the detector model on the held-out test dataset of the in-distribution, along with the associated OOD datasets (see Section 3.3.3). Finally, the evaluation program produces metrics for detection, classification (if applicable), and time efficiency—see Section 3.5. Finally, the metrics are aggregated over trials and datasets before being presented to the user.

The novel features of the benchmark include (1) randomly selected class sets for class-based tasks, (2) integrated dataset-based and class-based tasks (i.e. one user model is evaluated on both tasks), and (3) metrics for runtime, to measure efficiency. Although other components have been used before, our benchmark is also novel as a combination of: dataset-based and class-based tasks together, multiple training datasets, numerous OOD datasets, and three categories of metric.

Chapter 4

Investigation of TwinNet

In this chapter we present an investigation of a novel neural network architecture called *TwinNet*[1]. Specifically, we analyse the performance of TwinNet from the perspective of Out-of-Distribution (OOD) detection, and the potential advantage of ensembling this network compared to a regular network.

4.1 TwinNet architecture

Modern neural networks tend to encode a lot of redundancy. This is evident from the resilience of performance to dropout (Srivastava et al. 2014), and heavy "pruning" (Frankle and Carbin 2019). A natural question to ask in light of this is: what are the most important parts of a trained neural network? These parts could be parameters or functions. One way to approach this question is to take a well-performing, trained neural network A and a second, randomly initialised network B with identical architecture. We can then perform a sensitivity analysis by setting individual parts of the random network to the corresponding parts in the trained network.

A key part of deep neural networks is clearly nonlinear activation functions such as ReLU, because these enable universal approximation (Sonoda and Murata 2017). However, activation functions themselves are a fixed part of the architecture, so do not differentiate A and B. The activations out of a ReLU function *do* differentiate A and B, but copying those would make it nearly trivial for B to mimic A, provided the final activation layer is followed by a linear layer that can be trained. A middle ground can be found by decomposing the ReLU layer function from $\max(0,h)$ to $h.H(h)$, where

[1]Credit for the TwinNet architecture goes to Amos Storkey, with initial experiments done by Joseph Mellor. However, we are the first to ensemble TwinNet predictions and use it for OOD detection.

H is the Heaviside step function. The $H(h)$ can be seen as a mask that multiplies the h. We can then take the h from one network, the $H(h)$ from the other network, and multiply them together. This is the key idea of TwinNet: it changes the ReLU layers of B from $\max(0, h_B)$ to $h_B.H(h_A)$. Note that the hidden activations h_A and h_B must be derived from the same input x, so it is most efficient to execute A and B in tandem.

Now assume that A and B are image classifiers with ReLU activations and a linear output layer. The hybrid operation $h_B.H(h_A)$ would clearly deviate from the original functionality of A, given that B is otherwise random. It is therefore sensible to at least train the final linear layer of B. Unsurprisingly this is not sufficient for B to perform well. For example, using a ResNet-110 pretrained on CIFAR-10 for A and training the final layer of B for 10 epochs resulted in only **17.17%** accuracy in one experiment. One simple adjustment is enough to completely change the performance: copying the batch normalisation parameters of A. Doing so brought the accuracy up to **95.07%**.

4.2 The potential of TwinNet for robustness

The nearly as-good performance of B is remarkable, and demonstrates the importance of the learned mask $H(h)$. The fact that this encodes a great deal of useful information is perhaps not surprising. However, we find it surprising that using $H(h_A)$ (with batch normalisation) is sufficient for good performance, despite being modulated by the random parameters of B.

Of course, this result may not be practically useful. B seems unnecessary if we already have A, which as far as we know is more accurate. TwinNet is also not competitive on forward pass time, as it requires running both A and B. The grouping of their convolution and batch normalisation channels into shared layers, along with their identical shared parts, gives a modest speed-up compared to two copies of A, but this is still far from the speed of A alone. One potential advantage, however, is that the training time required for B can be less than 1/5th of A—in our experiments, 100 versus 10 epochs, with the 10 epochs taking almost double time.

Given the good performance and short training time of B, we wondered how much TwinNet benefits from ensembling, not just for accuracy but for robustness On one hand, B inherits a great deal of bias from A; on the other hand, most of the parameters of B are randomly initialised and A can be seen as a special instance in the space of possible B networks. It is therefore unclear how much variance is present between instances of B. Given the definite gain of training efficiency, and potential new insights, we felt

this was important to investigate. In the following sections we present an exploratory analysis of TwinNet's robustness.

4.3 Confidence and OOD detection

In a preliminary experiment on TwinNet, we used a pretrained 40-2 Wide ResNet from our Maximum Softmax Probability (MSP) baseline as network A (or "plain"). This network was trained for 100 epochs on five CIFAR-10 classes (indices 8, 2, 5, 6, 3), achieving a test error of 2.48%. We then constructed a TwinNet from A, training the final linear layer of B for 10 epochs; the settings were otherwise the same as A. Further details of the TwinNet implementation are given in Appendix B. Network B (or "twin") achieved a test error of 3.08%, corroborating the original ResNet-110 result. We then looked at the distribution of confidence on the CIFAR-10 test set for A and B on their respective correct and incorrect predictions, as shown in Fig. 4.1. As a control, we also reinitialised the final linear layer of A and retrained it for 10 epochs ("linear"), which achieved 2.56% test error.

Network A had a mean confidence of 99.4% on correct samples and 85.2% on incorrect samples, detecting the latter well with 94.9% AUROC. The control had a very similar distribution, though achieved slightly better error detection at 95.8% AUROC. Meanwhile, B differed drastically. In particular, its confidence was much lower on average: 95.4% for correct and only 66.5% for incorrect. Note that the difference in confidence was *disproportionate* to the difference in test error: we found that when A surpassed the test error of B in its original training run, its mean confidence was already over 99%. Curiously, the difference had little effect on error detection, with an AUROC of 94.8%. The confidence of B was also much more spread out, with double the standard deviation: 9.8% vs. 4.9%.

Next, we compared the OOD detection performance of A and B, using our OOD validation set. Fig. 4.1 illustrates the distribution of MSP scores for each dataset. We see the same general pattern of lower and more spread-out confidence for B. However, OOD detection turned out worse for B, achieving an AP of 56.1% compared to 59.0% for A. Examining the AP for each dataset, we see that the difference is nuanced: for example, A is much better at detecting uniform noise (64.8% vs. 35.5%), but worse at detecting averaged pairs of images (46.5% vs. 50.4%). We cannot draw strong conclusions here given this is only a single instance of the networks. The results are consistent with B being slightly worse than A on average, like for test error. However, the fact that a B

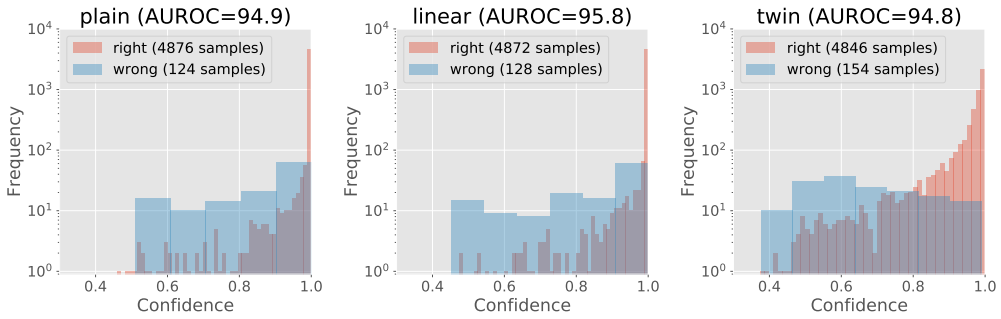

Figure 4.1: Distribution of confidence (maximum softmax probability) over the CIFAR-10 test set, split by correct and incorrect samples. AUROC is reported for detecting the incorrect samples.

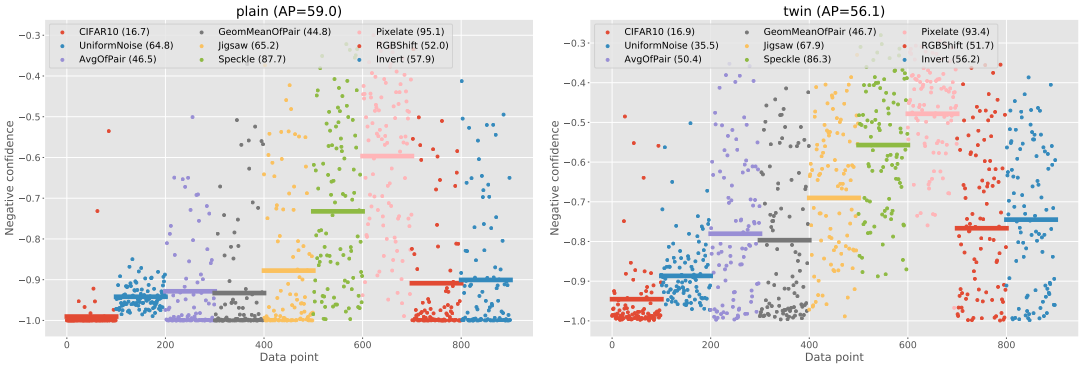

Figure 4.2: Distribution of negative maximum softmax probability over the CIFAR-10 test set and OOD validation set. The confidence of TwinNet A (left) is higher and more concentrated than TwinNet B (right), and does slightly better at OOD detection.

network *can* achieve similar OOD detection performance *and* has different comparative advantage to A supports the idea that B networks benefit significantly from ensembling.

4.4 Ensembling TwinNet

We considered two approaches to ensembling TwinNet. The first extended the idea of combining two networks in one module to N networks, with one A and multiple B. This meant that each B was trained on the same sequence of training data simultaneously, so we called this method "sim". The second approach trained multiple single-B TwinNets separately, with independently shuffled training batches, so we called this method "sep". Since "sep" requires a copy of A for each B, it takes more time than "sim" to train in series but less in parallel (unless "sim" is further parallelised at a sub-network level). For both approaches, the detection score was the MSP of the mean softmax over the B

Method	AUROC ↑	AP ↑	FPR95 ↓
plain	85.8 (0.1)	55.1 (0.1)	45.5 (0.2)
twin_sim_2	84.8 (0.0)	54.1 (0.1)	45.2 (0.2)
twin_sep_2	84.7 (0.1)	54.0 (0.1)	45.4 (0.2)
twin_sim_4	85.5 (0.0)	54.9 (0.1)	43.7 (0.1)
twin_sep_4	85.8 (0.0)	55.1 (0.1)	43.0 (0.2)
twin_sim_8	85.5 (0.1)	55.8 (0.1)	43.5 (0.3)
twin_sep_8	**86.4 (0.0)**	**56.3 (0.1)**	**41.2 (0.2)**

Table 4.1: Comparison of TwinNet B ensembles trained on the same sequence of training data ("sim") and independently shuffled sequences ("sep") for aggregated OOD detection on the validation set. The number appended to a method indicates the ensemble size. The reported values are the mean and standard deviation over 10 trials.

networks.

Table 4.1 compares the OOD detection of the A networks ("plain") to the "sim" and "sep" ensembles of B, with a varied ensemble size. We see that for up to 4 networks, B ensembles are unable to surpass A on Area Under the Receiver Operating Characteristic (AUROC) and Average Precision (AP). However, ensembling immediately surpasses A on False Positive Rate at true positive rate 95% (FPR95) and reaches up to 4.3% lower. For the largest size of 8, "sim" marginally surpasses A overall (e.g. +0.7% AP), while "sep" makes a bigger improvement on all metrics (e.g. +1.2% on AP). Generally, "sep" is better at detection than "sim" for a given ensemble size. This is unsurprising, as independent sequences of training data introduce more variance between each network's prediction. At inference time, we intended for "sep" networks to be combined in the same way as "sim" to achieve the same efficiency, but this is left for future work. For now, note that in principle the speed of "sim" and the performance of "sep" could be achieved simultaneously.

Having established that ensembling TwinNet can improve OOD detection from the baseline, albeit above a certain ensemble size, we proceeded to examine the trade-off between test error and OOD detection for the "sep" version. Figure 4.3 illustrates this trade-off. We see that, even for an ensemble of 10, B is unable to improve upon the test error of A. However, B does consistently reduce its test error with ensemble size. The same is almost true of the detection metrics—curiously the ensemble of 10 is comparable or slightly worse than 8. An ensemble size of 8 is therefore the

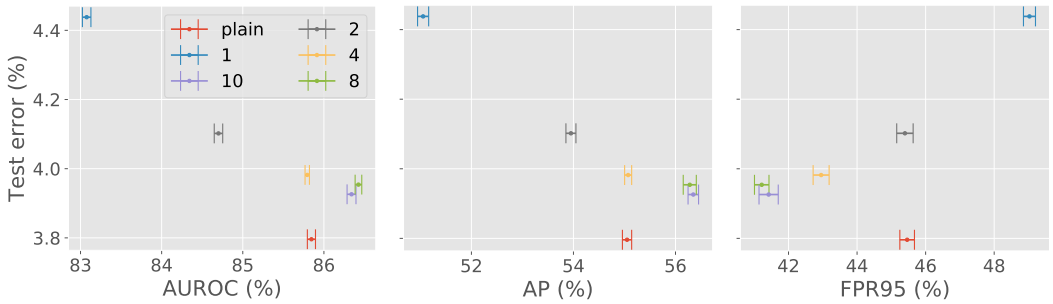

Figure 4.3: Trade-off between test error and OOD detection (on the validation set) for TwinNet B ensembles. Numbers in the legend indicated ensemble size. Error bars indicate standard deviation over 10 trials.

pareto-optimum for test error and OOD detection, out of the sizes tested here. We use this ensemble size in our final benchmark evaluation. We emphasise that once the cost of training the TwinNet model is sunk, there is practically no trade-off. Running B requires running A, so it is trivial to simultaneously use the more accurate predictions of A for classification, and the more robust predictions of the B ensemble for OOD detection.

In summary, we have seem that a single TwinNet can surpass a plain network at OOD detection on some datasets, but is significantly worse overall. By ensembling enough TwinNet models (8 in our experiment), we can surpass a plain network and achieve a pareto-optimum between test error and OOD detection. This was a preliminary investigation that both validated the potential benefit of TwinNet for robustness, and informed design decisions for our final evaluation. Our aim was to tune the design for OOD detection performance. We did *not* actively select for other important metrics, namely runtime and calibration. Those metrics measure side-effects of design decisions, and are covered in Chapter 5.

Chapter 5

Benchmark experiments

In this chapter we present and discuss the results of running our benchmark, following the procedure from Section 3.7. We evaluated the channel-wise Gaussian mixture model (GMM) (Section 3.2.1), the baselines explained in Section 3.4 (Maximum Softmax Probability (MSP), Monte Carlo Dropout (MCD), Random Prior (RP), RotNet) and the ensemble of 8 independently trained TwinNet models (Section 4). We only evaluated RP on CIFAR-10 for reasons explained near the end of Section 5.1, and we did not evaluate RotNet on TinyImageNet.

5.1 OOD detection

We first examine performance on the dataset-based Out-of-Distribution (OOD) detection tasks. Table 5.1 lists the detection metrics averaged first over the 10 random class subsets and then over the 10 trials for each subset. At this point it is worth reminding that the random-chance level is 50% for Area Under the Receiver Operating Characteristic (AUROC) and 95% for False Positive Rate at true positive rate 95% (FPR95). Meanwhile, the random-chance level for Average Precision (AP) is the relative frequency of positives or *skew*: $\approx 16.67\%$ for the dataset-based tasks and 50% for the class-based tasks. AP is therefore not directly comparable between the two tasks.

From Table 5.1 we see that the MSP baseline performs moderately well, in the sense that it greatly exceeds random-chance level in all cases. Its AUROC is also far better than GMM on CIFAR-10 and CIFAR-100. However, there is clearly much room for improvement upon MSP. The AP metrics highlight the inadequacy of MSP: accounting for skew makes it clear that MSP is imprecise, achieving only 58.1% for CIFAR-10.

We find that MCD is slightly worse than MSP across all metrics for dataset-based

tasks. This suggests that MCD is not particularly useful for detecting data from unfamiliar sources. Meanwhile, the TwinNet ensemble is also worse than both MSP and MCD overall. This is surprising, at least for CIFAR-10, because validation results suggested the TwinNet method would perform better (see Section 4.4). This disparity between validation and evaluation results highlights the challenge of designing an informative validation set. In general, benchmark users should be aware that fine-grained comparison of OOD detection performance may not generalise.

On the other hand, we find RotNet to be very effective at detection. For example, RotNet improves the MSP baseline from 58.1% AP to 91.2% AP on CIFAR-10. The reduction in FPR95 from 32.6% to 11.8% is also remarkable. These results lend further support to the conclusions of Hendrycks, Mazeika, Kadavath, et al. (2019) and Ahmed and Courville (2019). Namely, the representations learned via auxilliary tasks, such as predicting image rotation, can not only accelerate learning of the main classification task but also improve robustness. This makes auxilliary tasks very appealing all-round.

Almost all of the Deep Neural Network (DNN)-based detectors perform far above random-chance level, and RotNet is by far the best detector. However, comparing the results across datasets highlights a serious limitation: robustness to scale. For example, MSP goes from 58.1% AP on CIFAR-10, to 41.6% on CIFAR-100, to 28.6% AP on Tiny ImageNet. RotNet is no more robust to scale, suffering an enormous drop of 31.6% AP from CIFAR-10 to CIFAR-100. MCD and TwinNet suffer similar drops in performance, with TwinNet almost at random-chance level on Tiny ImageNet.

Meanwhile, though it performs worse overall, the GMM is more robust going from CIFAR-10 to CIFAR-100, most likely because the input dimension is the same and the GMM does not use class information. Note however that the *training* of this particular GMM model, with its full maximum-likelihood covariance estimation, scales poorly with dimension as d^2. This makes it infeasible going much beyond the dimensions of TinyImageNet. Indeed, the GMM drops drastically in performance along with the DNN methods when input dimension increases.

Results for the class-based tasks are listed in Table 5.2. Broadly the results tell a similar story, with notable differences. Comparing AUROC between Table 5.1 and 5.2 shows that the GMM performs much worse, around random-chance level. For CIFAR-10 and CIFAR-100, the DNN-based methods also perform significantly worse. This suggests that the class-based task is generally more difficult than the dataset-based task. Notably, performance does not degrade as much on Tiny ImageNet as for the dataset-based task. We attribute this partly to the challenging OOD datasets of

Table 5.1: Aggregate OOD detection metrics for the dataset-based tasks. For AUROC and AP, all values have standard deviation 0.1 or less, unless otherwise indicated in brackets. Arrows indicate the direction of improvement for the metric.

	GMM	MSP	RP	MCD	RotNet	TwinNet
			AUROC ↑			
CIFAR10	64.6	88.4	50.3	86.8	97.5	86.8
CIFAR100	66.3	79.5	-	77.3	87.9	78.7
TinyImageNet	33.2	67.4	-	65.0	-	53.8
			AP ↑			
CIFAR10	52.5	58.1	27.8	53.5	91.2	55.7 (0.2)
CIFAR100	52.5	41.6	-	38.5	59.6	37.9
TinyImageNet	25.5	28.6	-	27.0	-	19.0
			FPR95 ↓			
CIFAR10	61.1	32.6 (0.2)	76.5	33.1 (0.4)	11.8 (0.2)	35.4 (0.3)
CIFAR100	58.9	47.7 (0.2)	-	50.6 (0.2)	35.9 (0.3)	50.2 (0.2)
TinyImageNet	74.0	64.5 (0.2)	-	67.0	-	76.8

ImageNet-x and ImageNet-O in the latter task, with ImageNet-O curated especially to fool DNN classifiers (Hendrycks, Zhao, et al. 2020). MSP achieves only 75.2% and 68.5% AUROC on these datasets, respectively—see Table C.7. What is more surprising is that Tiny ImageNet classifiers are worse than random chance at detecting noise, at 45.0% AUROC for Gaussian and 42.9% AUROC for Rademacher. This indicates a major challenge for DNN models on large-scale datasets.

Another notable departure of class-based task results is that the TwinNet ensemble performs slightly better than MSP on CIFAR-10, rather than slightly worse. However, it still scales poorly with data dimensions, becoming much worse again on Tiny ImageNet. Meanwhile, MCD is slightly better than MSP across the datasets and metrics. Ultimately, RotNet maintains a strong lead on class-based tasks, but like all the other methods, it suffers from a poor false-positive rate.

The one method we have overlooked so far is RP. As Table 5.1 and 5.2 show, RP performed very poorly on CIFAR-10, either not far above or worse than random chance levels. Due to this very poor performance, we did not proceed to evaluate RP on CIFAR-

Table 5.2: Aggregate OOD detection metrics for the class-based tasks.

	GMM	MSP	RP	MCD	RotNet	TwinNet
			AUROC ↑			
CIFAR10	49.1	82.9	45.2	83.4	88.4	83.5
CIFAR100	51.0	76.3	-	76.4	78.6	74.7
TinyImageNet	49.4	72.6	-	73.1	-	64.5
			AP ↑			
CIFAR10	50.0	80.7	46.9	81.0	88.8	81.5
CIFAR100	51.2	72.2	-	72.4	75.5	70.6
TinyImageNet	49.5	67.5	-	68.0	-	60.1
			FPR95 ↓			
CIFAR10	94.3	62.8	95.7	57.4	51.8	60.1
CIFAR100	95.0	66.0	-	65.6	65.4	70.7
TinyImageNet	95.0	66.6	-	66.3	-	79.0

100 or Tiny ImageNet. To understand the poor performance, we refer to Figure 5.1. This compares the distribution of RP uncertainty estimates for the datasets evaluated in Ciosek et al. (2019). We see that all of the uncertainties for non-training datasets, *including* the in-distribution test set, are distributed far away from the uncertainties on the training dataset. Furthermore, the uncertainties for the in-distribution test set are in a very similar range to the OOD datasets. This shows that RP is very effective at detecting data outside of the training set, but ineffective at distinguishing new in-distribution test points from OOD test points. In practice, we assume that an OOD detector observes test inputs sequentially, and must distinguish in-distribution test inputs from OOD test inputs one-at-a-time. We therefore find that RP is not useful for practical OOD detection. Essentially, the problem is that RP is too conservative. However, this does not necessarily detract from RP as a novelty detection or uncertainty estimation method in certain domains.

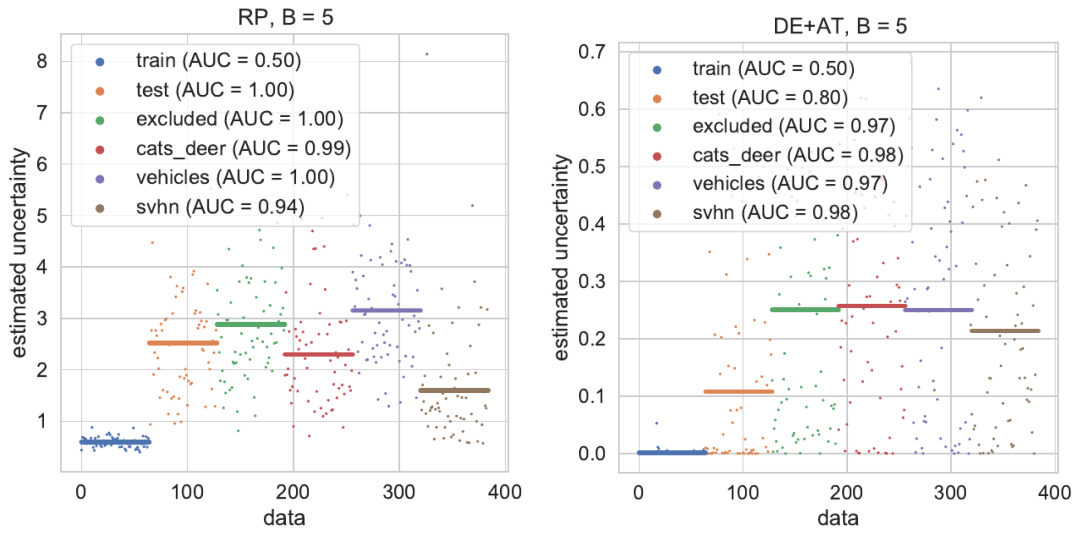

Figure 5.1: Distribution of uncertainties for RP (left) versus Deep Ensemble with Adversarial Training (right), using an ensemble size of 5. Replicated from Ciosek et al. (2019).

5.2 Classification, error detection and calibration

We now look at metrics related to predictive quality of classifiers. Note that only MSP, MCD, RotNet and TwinNet have their own classifier. Note also that, while the TwinNet results here are for the B ensemble, predictive performance was consistently worse than the A network and in practice A should be used for classification instead. Finally, note that MCD did not require additional training and only modified predictions at inference time.

Table 5.3 lists the test error for each predictive model on each in-distribution test set, averaged over the 10 class subsets for each in-distribution. We find that MCD is comparable or slightly better than MSP at classification. Meanwhile, RotNet is far better on CIFAR-10 and CIFAR-100, demonstrating an additional advantage to rotation prediction in our training setting. Meanwhile, the TwinNet ensemble is slightly worse on CIFAR-10, and like OOD detection, it degrades severely for CIFAR-100 and Tiny ImageNet.

Table 5.4 shows the AUROC scores measuring a classifier's ability to detect its own errors using the (averaged) MSP score. Again, MCD is slightly better than MSP in this regard. It is interesting that RotNet is instead worse than MSP and even TwinNet here, despite being much better at classification. This may be explained by a correlation between the number of incorrect samples and the difficulty of detecting those samples.

Table 5.3: Test error (%)

	MSP	MCD	RotNet	TwinNet
CIFAR10	3.80	3.81	2.95	3.95
CIFAR100	18.58	18.47	17.91	20.69
TinyImageNet	31.83	31.40	-	45.94

Table 5.4: Error detection AUROC (%)

	MSP	MCD	RotNet	TwinNet
CIFAR10	93.8	94.3	90.0	93.2
CIFAR100	88.8	89.0	83.2	86.3
TinyImageNet	86.5	86.6	-	79.5

Finally, Table 5.5 presents the average Root Mean Square (RMS) calibration error of our classifiers. MSP proves to be moderately well-calibrated, but there is still much room for improvement, especially for the larger-scale datasets. Meanwhile, MCD greatly reduces calibration error, and RotNet is similar overall to MSP. TwinNet proves to be second-best on CIFAR-10 here, but once again scales very poorly. Overall, the results here show that RotNet is the most effective at classification, but MCD is not far behind and is more effective than RotNet as an in-distribution uncertainty estimate.

5.3 Time

Finally, to assess the relative efficiency of methods we examine inference and scoring time. Inference times are listed in the top section of Table 5.6. Comparing MCD, RotNet and TwinNet to the MSP baseline, we find that RotNet bears a much lower inference

Table 5.5: RMS Calibration Error (%)

	MSP	MCD	RotNet	TwinNet
CIFAR10	5.3	3.1	4.6	4.0
CIFAR100	13.2	4.9	13.3	29.8
TinyImageNet	14.7	5.7	-	31.2

Table 5.6: Inference time in milliseconds per batch of 100 images and OOD score time in milliseconds per dataset, averaged over all datasets. Where a method does not have a predictive model directly associated, we report the MSP inference time.

	GMM	MSP	RP	MCD	RotNet	TwinNet
			Inference			
CIFAR10	10.7	10.7	10.7	872.7	22.3	543.9
CIFAR100	10.9	10.9	-	875.7	21.9	543.4
TinyImageNet	10.6	10.6	-	3279.9	-	2054.6
			Score			
CIFAR10	1630	0.1	1240	0.2	11.1	0.2
CIFAR100	1700	0.1	-	0.3	12.3	0.3
TinyImageNet	4450	0.1	-	0.6	-	0.4

cost, though it is still about double the time due to being run on four orientations per input. We expect that most applications could tolerate this additional cost given the large gains in performance noted in the previous sections. Meanwhile, MCD and TwinNet are much more costly, at an order of magnitude above MSP.

The high cost of MSP and TwinNet comes with important caveats. Firstly, we selected the number of samples for MCD partly so that it took a comparable time to TwinNet. We did so because MCD is the most natural baseline for TwinNet of the methods we considered, so we avoided giving TwinNet an unfair advantage by having more time. The second caveat is that the TwinNet ensemble was implemented as 8 separate modules, when it could have been more efficiently implemented as a single module with group convolutions. The samples of MCD and TwinNet could also have been parallelised, reducing the execution time by a factor of up to 20 and 8, respectively. We expect that these efficiency gains would bring MCD and TwinNet into the same order of magnitude as MSP and RotNet, but still significantly worse.

As for scoring efficiency, the bottom section of Table 5.6 also shows large variance between methods. The fastest is again MSP, closely followed by MCD and TwinNet on the order of 0.1 milliseconds. Unlike inference, the RotNet score takes much longer to compute, but still does not seem to be a limiting factor. However, the GMM and RP both bear significantly greater cost to score, due to using separate models from the classifier.

Chapter 6

Conclusions

This thesis was motivated by the failure of Deep Neural Network (DNN) classifiers to handle Out-of-Distribution (OOD) inputs. The failure is particularly problematic for safety-critical applications such as medical imaging and autonomous driving. To improve the trustworthiness of DNN classifiers, we focused on OOD detection: using a scoring model to detect when the input to a classifier is OOD, so that this input can be handled safely and reliably.

Despite a considerable body of work on OOD detection for DNNs in the literature, we identified several limitations to evaluation. These include evaluating only one training dataset, lack of diversity in OOD datasets, using the same OOD distribution in hyperparameter tuning and testing, and neglecting efficiency as a measure of performance. We proposed a new benchmark that addresses many of the limitations. The the benchmark includes a novel task design, using randomly selected class sets and integrating dataset-based and class-based tasks. It is also more well-rounded than most other benchmarks, using multiple training datasets, numerous OOD datasets, and three categories of metric. Besides designing the benchmark, we investigated the potential merit of ensembling an architecture called TwinNet, as a new OOD detection method, before evaluating this and other baselines on our benchmark. The evaluation makes a larger-scale assessment of the Random Prior (RP) (Ciosek et al. 2019) and RotNet (Hendrycks, Mazeika, and Dietterich 2019) methods compared to previous work.

The results of the benchmark showed that both dataset-based and class-based OOD detection tasks are difficult relative to the standard we expect for trustworthy machine learning, with much room for improvement. Furthermore, a simple Gaussian mixture model (GMM) baseline was inadequate at these tasks and inefficient at inference time, providing some evidence that DNN classifiers are marginally useful over purely unsuper-

vised or density-estimation methods. However, a major issue with DNN classifier-based methods is their robustness to scale, as detection performance degraded significantly with larger input and output dimensions for all of those methods.

We also observed some trade-offs between OOD detection tasks: for example, Monte Carlo Dropout (MCD) was slightly worse than Maximum Softmax Probability (MSP) at dataset-based tasks but slightly better at class-based tasks. It was also much better than any classifier method for uncertainty estimation on the in-distribution. Meanwhile, the TwinNet ensemble was generally disappointing, usually being worse than the MSP baseline and degrading the most with input and output dimensions. Overall, adding rotation prediction as an auxilliary task (RotNet) was by far the best at OOD detection, best on classification error, and competitive on calibration and efficiency metrics.

The benchmark results suggest several directions for future work. Firstly, although TwinNet was generally disappointing, we believe our investigation in Chapter 4 still shows some potential for combining competitive classification performance with very efficient training. We are excited to see further investigation to understand how it might be used better. Secondly, given the excellent performance of RotNet, other auxilliary tasks could be explored further. For example, we could test whether the benefit of optimised views for contrastive learning (Tian et al. 2020) transfers to OOD detection. We could also combine RotNet with Outlier Exposure (Hendrycks, Mazeika, and Dietterich 2019). Indeed, MCD, RotNet and TwinNet could all be combined in different ways, as their scoring models can complement each other.

In terms of benchmark design, while we focused on generic datasets for object recognition, it would be valuable to focus on specific application domains to understand how benchmarks need to be adapted, and whether the relative performance of the methods generalises. It would also be useful to compare more efficient and competitive unsupervised baselines. Finally, given that performance on the validation set did not always transfer well to evaluation, it is important to investigate the design of fair yet informative validation data.

Bibliography

Ahmed, Faruk and Aaron Courville (2019). "Detecting semantic anomalies". In: *arXiv: 1908.04388 [cs]*. arXiv: 1908.04388. URL: http://arxiv.org/abs/1908.04388.

Burda, Yuri et al. (2018). "Exploration by Random Network Distillation". In: *arXiv: 1810.12894 [cs, stat]*. arXiv: 1810.12894. URL: http://arxiv.org/abs/1810.12894.

Cao, Tianshi et al. (2020). "A Benchmark of Medical Out of Distribution Detection". In: *arXiv:2007.04250 [cs, stat]*. arXiv: 2007.04250. URL: http://arxiv.org/abs/2007.04250.

Cimpoi, Mircea et al. (2014). "Describing Textures in the Wild". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3606–3613. URL: https://openaccess.thecvf.com/content_cvpr_2014/html/Cimpoi_Describing_Textures_in_2014_CVPR_paper.html.

Ciosek, Kamil et al. (2019). "Conservative Uncertainty Estimation By Fitting Prior Networks". In: International Conference on Learning Representations. URL: https://openreview.net/forum?id=BJlahxHYDS.

Davis, Jesse and Mark Goadrich (2006). "The relationship between Precision-Recall and ROC curves". In: *Proceedings of the 23rd international conference on Machine learning - ICML '06*. the 23rd international conference. Pittsburgh, Pennsylvania: ACM Press, pp. 233–240. ISBN: 978-1-59593-383-6. DOI: 10.1145/1143844.1143874. URL: http://portal.acm.org/citation.cfm?doid=1143844.1143874.

Emmott, Andrew et al. (2016). "A Meta-Analysis of the Anomaly Detection Problem". In: *arXiv:1503.01158 [cs, stat]*. arXiv: 1503.01158. URL: http://arxiv.org/abs/1503.01158.

Frankle, Jonathan and Michael Carbin (2019). "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *ICLR*.

Gal, Yarin and Zoubin Ghahramani (2016). "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *International Conference on Machine Learning*. International Conference on Machine Learning, pp. 1050–1059. URL: http://proceedings.mlr.press/v48/gal16.html.

Guo, Chuan et al. (2017). "On calibration of modern neural networks". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, pp. 1321–1330.

Hein, Matthias, Maksym Andriushchenko, and Julian Bitterwolf (2019). "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, pp. 41–50. URL: http://openaccess.thecvf.com/content_CVPR_2019/html/Hein_Why_ReLU_Networks_Yield_High-Confidence_Predictions_Far_Away_From_the_CVPR_2019_paper.html.

Hendrycks, Dan and Kevin Gimpel (2016). "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: URL: https://openreview.net/forum?id=Hkg4TI9xl.

Hendrycks, Dan, Mantas Mazeika, and Thomas Dietterich (2019). "Deep Anomaly Detection with Outlier Exposure". In: *arXiv:1812.04606 [cs, stat]*. URL: http://arxiv.org/abs/1812.04606.

Hendrycks, Dan, Mantas Mazeika, Saurav Kadavath, et al. (2019). "Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., pp. 15663–15674. URL: http://papers.nips.cc/paper/9697-using-self-supervised-learning-can-improve-model-robustness-and-uncertainty.pdf.

Hendrycks, Dan, Kevin Zhao, et al. (2020). "Natural Adversarial Examples". In: *arXiv:1907.07174 [cs, stat]*. arXiv: 1907.07174. URL: http://arxiv.org/abs/1907.07174.

Komodakis, Nikos (2018). "Unsupervised Representation Learning by Predicting Image Rotations". In: International Conference on Learning Representations. URL: https://openreview.net/forum?id=S1v4N2l0-.

Krizhevsky, Alex (2009). *Learning Multiple Layers of Features from Tiny Images*, p. 60.

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell (2017). "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates,

Inc., pp. 6402–6413. URL: http://papers.nips.cc/paper/7219-simple-and-scalable-predictive-uncertainty-estimation-using-deep-ensembles.pdf.

Lee, Kimin et al. (2018a). "Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples". In: *arXiv:1711.09325 [cs, stat]*. URL: http://arxiv.org/abs/1711.09325.

— (2018b). "Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples". In: International Conference on Learning Representations. URL: https://openreview.net/forum?id=ryiAv2xAZ&noteId=ryiAv2xAZ.

Leibig, Christian et al. (2017). "Leveraging uncertainty information from deep neural networks for disease detection". In: *Scientific Reports* 7.1, pp. 1–14. ISSN: 2045-2322. DOI: 10.1038/s41598-017-17876-z. URL: https://www.nature.com/articles/s41598-017-17876-z.

Liang, Shiyu, Yixuan Li, and R. Srikant (2018). "Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks". In: *arXiv:1706.02690 [cs, stat]*. URL: http://arxiv.org/abs/1706.02690.

Meinke, Alexander and Matthias Hein (2020). "Towards neural networks that provably know when they don't know". In: *arXiv:1909.12180 [cs, stat]*. URL: http://arxiv.org/abs/1909.12180.

Nalisnick, Eric et al. (2019). "Do Deep Generative Models Know What They Don't Know?" In: *arXiv:1810.09136 [cs, stat]*. arXiv: 1810.09136. URL: http://arxiv.org/abs/1810.09136.

Netzer, Yuval et al. (2011). "Reading Digits in Natural Images with Unsupervised Feature Learning". In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. URL: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Rabanser, Stephan, Stephan Günnemann, and Zachary Lipton (2019). "Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., pp. 1396–1408. URL: http://papers.nips.cc/paper/8420-failing-loudly-an-empirical-study-of-methods-for-detecting-dataset-shift.pdf.

Russakovsky, Olga et al. (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *arXiv:1409.0575 [cs]*. arXiv: 1409.0575. URL: http://arxiv.org/abs/1409.0575.

Schölkopf, Bernhard et al. (2000). "Support Vector Method for Novelty Detection". In: *Advances in Neural Information Processing Systems 12*. Ed. by S. A. Solla, T. K. Leen, and K. Müller. MIT Press, pp. 582–588. URL: http://papers.nips.cc/paper/1723-support-vector-method-for-novelty-detection.pdf.

Shafaei, Alireza (2019). "A Less Biased Evaluation of Out-of-distribution Sample Detectors". In: p. 13.

Sonoda, Sho and Noboru Murata (2017). "Neural network with unbounded activation functions is universal approximator". In: *Applied and Computational Harmonic Analysis* 43.2, pp. 233–268. ISSN: 1063-5203. DOI: 10.1016/j.acha.2015.12.005. URL: http://www.sciencedirect.com/science/article/pii/S1063520315001748.

Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958. ISSN: 1532-4435.

Tian, Yonglong et al. (2020). "What makes for good views for contrastive learning". In: *arXiv:2005.10243 [cs]*. arXiv: 2005.10243. URL: http://arxiv.org/abs/2005.10243.

Torralba, Antonio, Rob Fergus, and William T. Freeman (2008). "80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.11, pp. 1958–1970. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2008.128.

Xiao, Han, Kashif Rasul, and Roland Vollgraf (2017). "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: *arXiv:1708.07747 [cs, stat]*. arXiv: 1708.07747. URL: http://arxiv.org/abs/1708.07747.

Yu, Fisher et al. (2016). "LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop". In: *arXiv:1506.03365 [cs]*. arXiv: 1506.03365. URL: http://arxiv.org/abs/1506.03365.

Zagoruyko, Sergey and Nikos Komodakis (2017). "Wide Residual Networks". In: *arXiv:1605.07146 [cs]*. arXiv: 1605.07146. URL: http://arxiv.org/abs/1605.07146.

Appendix A

Supplementary figures

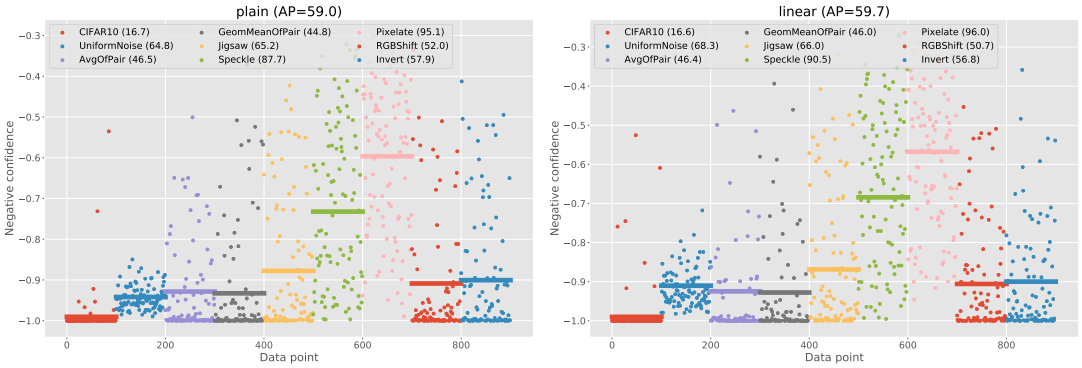

Figure A.1: Distribution of negative maximum softmax probability over the CIFAR-10 test set and OOD validation set for the plain network, and the same network with the final linear layer reinitialised and trained for 10 epochs. The distributions and performance are very similar, unlike Fig. 4.1.

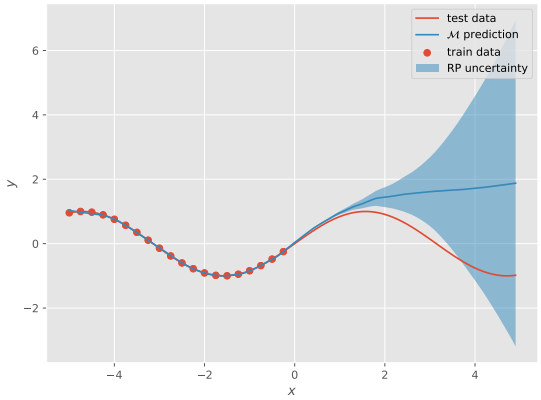

Figure A.2: 2D regression example of the RP method. The supervised model $\mathcal{M}$, a multilayer perceptron with ReLU activations, is trained to predict a sine function. Meanwhile, a learner is trained to mimic a random prior on the same data. The Mean Squared Error (MSE) between the learner and prior (shaded) provides an apparently reasonable estimate of variance.

Appendix B

Implementation detail

B.1 Supervised model

For all experiments, the supervised model was a 40-2 Wide Residual Network (Zagoruyko and Komodakis 2017). We trained this using cross-entropy loss for 100 epochs with batch size 128 on a single Tesla K40c GPU. The optimizer was stochastic gradient descent with initial learning rate 0.1, L2 weight regularisation of 0.0005, Nesterov momentum 0.9, and a cosine annealing learning rate schedule ending at 10^{-5}. We applied data augmentations of random crop and random horizontal flip on all samples. All of these settings were based on previous work, particularly Hendrycks, Mazeika, and Dietterich (2019), and not optimized ourselves. All computation related to models was implemented using the PyTorch library.

B.2 Random Prior

We set the prior scale to 1.0 based on validation results, selected from (0.1, 1.0, 2.0, 5.0).

B.3 Monte Carlo Dropout

Based on validation results we found the MSP score better than standard deviation on the mean prediction of Monte Carlo Dropout.

B.4 RotNet

We set λ to 0.5 based on Hendrycks, Mazeika, Kadavath, et al. (2019), without any tuning.

B.5 TwinNet

When implementing TwinNet, one must consider how B depends on hidden activations of A. This limits the ability to run A and B as separate modules in parallel—at best, some sophisticated message passing is required. We implemented TwinNet as a single network that combines A and B. In this module, the convolution and batch normalisation layers of A and B were grouped along the channel dimension. Then, the ReLU layers were modified as described in the main matter, to implement $h_B.H(h_A)$ for B. The final channels before the output layer were then partitioned by the number of classes and fed into separate linear layers.

Appendix C

Additional benchmark results

Table C.1: Dataset-based CIFAR-10 AUROC

	GMM	MCD	MSP	Rotation	RP	TwinNet
SVHN	1.4	89.9	91.6	99.2	25.7	89.7
CIFAR100	47.5	86.4	86.1	93.3	51.0	86.7
LSUN	79.9	89.7	90.2	95.1	58.0	91.5
GaussianNoise	100.0	84.2	89.1	99.4	25.5	80.2
RademacherNoise	100.0	83.6	84.9	99.5	98.6	85.2
Textures	58.6	87.0	88.5	98.5	42.9	87.5
mean	64.6	86.8	88.4	97.5	50.3	86.8

Table C.2: Dataset-based CIFAR-10 AP

	GMM	MCD	MSP	Rotation	RP	TwinNet
SVHN	8.9	56.4	62.4	96.5	11.0	59.0
CIFAR100	17.1	53.8	54.1	79.9	20.1	56.2
LSUN	46.8	60.0	61.4	84.0	19.3	65.1
GaussianNoise	100.0	45.9	54.9	95.8	11.7	43.7
RademacherNoise	100.0	51.6	56.8	96.4	84.8	52.8
Textures	42.5	53.5	59.2	94.7	20.2	57.2
mean	52.5	53.5	58.1	91.2	27.8	55.7

Table C.3: Dataset-based CIFAR-10 FPR95

	GMM	MCD	MSP	Rotation	RP	TwinNet
SVHN	100.0	28.3	26.2	3.9	97.9	33.3
CIFAR100	97.5	47.9	53.3	33.6	95.4	50.8
LSUN	69.8	32.6	33.0	23.3	86.2	28.5
GaussianNoise	0.0	23.3	18.0	1.7	80.1	30.7
RademacherNoise	0.0	24.4	22.2	1.5	1.7	23.5
Textures	99.6	42.1	43.1	7.0	97.7	45.6
mean	61.1	33.1	32.6	11.8	76.5	35.4

Table C.4: Dataset-based CIFAR-100 AUROC

	GMM	MCD	MSP	Rotation	RP	TwinNet
SVHN	2.1	82.4	84.7	95.6	-	80.9
CIFAR10	53.9	78.7	77.6	76.7	-	77.9
LSUN	81.9	76.7	76.9	77.9	-	74.8
GaussianNoise	100.0	73.8	78.4	93.7	-	79.8
RademacherNoise	100.0	73.5	77.8	93.6	-	81.0
Textures	59.7	78.7	81.5	90.1	-	77.8
mean	66.3	77.3	79.5	87.9	-	78.7

Table C.5: Dataset-based CIFAR-100 AP

	GMM	MCD	MSP	Rotation	RP	TwinNet
SVHN	8.9	44.5	48.2	80.9	-	38.6
CIFAR10	17.8	38.7	37.0	37.1	-	37.1
LSUN	46.7	35.4	36.0	38.6	-	32.3
GaussianNoise	100.0	30.3	34.7	67.2	-	36.2
RademacherNoise	100.0	42.4	48.6	66.4	-	47.4
Textures	41.5	40.0	45.1	67.2	-	36.0
mean	52.5	38.5	41.6	59.6	-	37.9

Table C.6: Dataset-based CIFAR-100 FPR95

	GMM	MCD	MSP	Rotation	RP	TwinNet
SVHN	100.0	47.1	41.0	16.7	-	50.9
CIFAR10	91.2	60.7	63.3	70.4	-	63.9
LSUN	63.0	61.7	63.9	66.3	-	65.2
GaussianNoise	0.0	37.9	32.4	12.6	-	31.4
RademacherNoise	0.0	37.7	32.5	12.0	-	28.3
Textures	98.9	58.3	53.2	37.6	-	61.3
mean	58.9	50.6	47.7	35.9	-	50.2

Table C.7: Dataset-based TinyImageNet AUROC

	GMM	MCD	MSP	RotNet	RP	TwinNet
SVHN	0.1	79.3	81.8	-	-	68.6
LSUN	11.5	76.2	75.6	-	-	68.2
ImageNet	11.9	75.2	74.6	-	-	66.3
ImageNetO	12.3	68.5	69.5	-	-	59.0
GaussianNoise	80.9	45.0	50.7	-	-	30.6
RademacherNoise	100.0	42.9	49.4	-	-	28.1
Textures	15.5	67.7	70.1	-	-	55.6
mean	33.2	65.0	67.4	-	-	53.8

Table C.8: Dataset-based TinyImageNet AP

	GMM	MCD	MSP	RotNet	RP	TwinNet
SVHN	8.8	40.3	45.1	-	-	23.4
LSUN	9.5	33.5	33.0	-	-	25.5
ImageNet	9.4	32.7	32.2	-	-	24.2
ImageNetO	9.4	25.9	27.0	-	-	19.3
GaussianNoise	31.4	14.9	16.4	-	-	12.0
RademacherNoise	99.9	15.0	17.2	-	-	11.5
Textures	9.7	26.3	29.2	-	-	17.5
mean	25.5	27.0	28.6	-	-	19.0

Table C.9: Dataset-based TinyImageNet FPR95

	GMM	MCD	MSP	RotNet	RP	TwinNet
SVHN	100.0	53.6	49.6	-	-	60.2
LSUN	98.4	60.4	61.0	-	-	71.8
ImageNet	99.3	62.6	63.2	-	-	75.3
ImageNetO	99.7	75.2	73.4	-	-	84.4
GaussianNoise	20.5	70.5	65.8	-	-	80.1
RademacherNoise	0.0	69.8	64.3	-	-	80.7
Textures	99.9	76.7	74.0	-	-	84.9
mean	74.0	67.0	64.5	-	-	76.8

Table C.10: Class-based CIFAR-10 AUROC

	GMM	MCD	MSP	RotNet	RP	TwinNet
0	42.5	86.4	85.9	93.5	40.4	85.1
1	44.9	81.0	80.5	88.8	35.8	79.2
2	38.6	83.4	82.0	92.0	46.6	83.4
3	54.7	83.4	82.3	84.6	43.0	81.3
4	44.5	88.8	87.5	96.7	51.4	90.0
5	60.4	83.0	81.8	85.9	47.1	82.7
6	45.2	86.1	86.3	86.7	45.2	86.8
7	45.0	85.8	85.4	89.6	35.9	84.8
8	47.6	83.1	82.7	75.7	47.0	84.9
9	67.3	73.4	74.2	90.6	59.7	76.8
mean	49.1	83.4	82.9	88.4	45.2	83.5

Table C.11: Class-based CIFAR-10 AP

	GMM	MCD	MSP	RotNet	RP	TwinNet
0	45.1	85.0	84.6	94.2	43.4	84.5
1	47.4	77.3	77.3	89.0	41.0	76.9
2	42.1	81.5	80.5	92.2	48.0	82.4
3	53.8	80.7	79.8	85.0	44.9	78.9
4	45.3	87.2	86.3	96.6	51.8	88.1
5	59.2	80.9	80.2	86.9	48.5	80.6
6	47.5	82.7	83.3	86.8	47.1	84.7
7	45.3	82.9	82.9	89.8	40.5	82.7
8	48.1	80.2	80.1	75.6	48.0	81.7
9	66.0	71.6	72.3	91.7	55.7	74.3
mean	50.0	81.0	80.7	88.8	46.9	81.5

Table C.12: Class-based CIFAR-10 FPR95

	GMM	MCD	MSP	RotNet	RP	TwinNet
0	96.3	52.4	58.8	36.8	96.5	66.3
1	97.1	59.6	65.8	48.1	98.0	71.4
2	98.1	66.2	75.4	38.2	95.1	67.3
3	92.4	55.8	60.2	68.4	95.5	64.7
4	96.1	48.7	57.4	14.7	95.1	41.5
5	89.0	63.4	69.7	70.9	96.8	61.6
6	96.6	39.9	44.8	58.5	96.0	47.8
7	94.7	50.9	55.2	48.9	97.1	56.4
8	96.0	57.6	61.6	78.9	96.1	50.8
9	86.5	80.0	79.2	54.9	90.3	72.8
mean	94.3	57.4	62.8	51.8	95.7	60.1

Table C.13: Class-based CIFAR-100 AUROC

	GMM	MCD	MSP	RotNet	RP	TwinNet
0	54.6	76.9	76.5	74.9	-	74.4
1	53.3	75.1	74.6	78.2	-	74.0
2	50.1	75.5	75.1	82.0	-	75.4
3	47.3	77.4	77.9	77.1	-	74.9
4	51.1	77.3	77.0	74.9	-	75.2
5	54.3	75.6	75.3	79.1	-	74.3
6	48.6	79.5	79.4	78.8	-	78.2
7	47.0	72.6	73.6	80.7	-	69.8
8	48.9	76.3	76.3	80.5	-	75.1
9	54.6	77.6	77.2	79.1	-	76.1
mean	51.0	76.4	76.3	78.6	-	74.7

Table C.14: Class-based CIFAR-100 AP

	GMM	MCD	MSP	RotNet	RP	TwinNet
0	53.5	72.7	72.4	71.2	-	70.0
1	52.7	71.3	70.7	75.4	-	69.2
2	51.5	72.1	71.4	79.2	-	72.0
3	48.3	73.9	74.2	74.1	-	71.2
4	52.4	73.9	73.1	72.5	-	71.5
5	52.6	71.2	71.1	76.3	-	69.8
6	50.2	75.8	75.7	75.6	-	74.0
7	48.2	68.2	69.4	76.9	-	66.3
8	49.6	71.6	71.4	77.2	-	70.4
9	53.4	73.4	72.8	76.3	-	71.8
mean	51.2	72.4	72.2	75.5	-	70.6

Table C.15: Class-based CIFAR-100 FPR95

	GMM	MCD	MSP	RotNet	RP	TwinNet
0	91.3	65.7	66.8	72.5	-	72.3
1	93.2	68.0	70.3	66.8	-	70.2
2	97.2	67.8	68.2	56.4	-	69.8
3	96.8	64.4	61.9	68.4	-	72.6
4	96.0	67.6	67.0	72.5	-	71.8
5	91.5	70.0	70.1	66.1	-	71.1
6	97.6	57.4	57.9	67.4	-	63.3
7	96.5	72.5	71.1	58.0	-	80.6
8	97.0	62.5	63.3	59.5	-	67.9
9	93.1	60.1	63.0	66.6	-	67.7
mean	95.0	65.6	66.0	65.4	-	70.7

Table C.16: Class-based Tiny ImageNet AUROC

	GMM	MCD	MSP	RotNet	RP	TwinNet
0	50.7	72.6	72.0	-	-	69.0
1	50.9	73.0	72.6	-	-	63.8
2	49.1	73.6	73.1	-	-	66.0
3	47.6	72.0	71.7	-	-	64.2
4	48.8	74.5	74.2	-	-	62.7
5	49.7	72.0	71.6	-	-	63.2
6	50.8	73.7	73.2	-	-	61.6
7	51.2	71.9	71.4	-	-	64.6
8	47.7	73.3	72.8	-	-	64.5
9	48.0	73.7	73.0	-	-	65.1
mean	49.4	73.1	72.6	-	-	64.5

Table C.17: Class-based Tiny ImageNet AP

	GMM	MCD	MSP	RotNet	RP	TwinNet
0	50.2	67.3	66.5	-	-	63.8
1	51.1	67.5	67.1	-	-	59.7
2	48.2	68.8	68.6	-	-	61.2
3	48.6	66.4	66.2	-	-	59.9
4	49.1	69.0	68.7	-	-	58.6
5	50.0	67.1	66.8	-	-	58.7
6	50.2	69.3	68.7	-	-	58.2
7	51.5	66.7	66.2	-	-	60.4
8	48.3	68.5	67.9	-	-	60.1
9	48.2	69.1	68.5	-	-	60.5
mean	49.5	68.0	67.5	-	-	60.1

Table C.18: Class-based Tiny ImageNet FPR95

	GMM	MCD	MSP	RotNet	RP	TwinNet
0	94.7	66.8	67.2	-	-	72.0
1	94.5	67.0	66.4	-	-	80.9
2	95.2	64.7	66.1	-	-	77.7
3	95.6	66.7	67.7	-	-	80.2
4	96.0	63.4	63.2	-	-	82.6
5	95.5	68.5	68.1	-	-	79.3
6	94.0	67.1	66.4	-	-	82.6
7	94.7	67.6	68.2	-	-	79.1
8	95.9	66.3	67.2	-	-	78.6
9	94.4	65.2	65.4	-	-	77.3
mean	95.0	66.3	66.6	-	-	79.0

Table C.19: Error prediction AP (%)

	MSP	MCD	RotNet	TwinNet
CIFAR10	39.1	39.4	26.9	38.5
CIFAR100	61.4	61.6	51.4	60.1
TinyImageNet	71.8	72.2	-	73.7

Table C.20: Error prediction FPR95 (%)

	MSP	MCD	RotNet	TwinNet
CIFAR10	25.0	21.8	50.2	26.9
CIFAR100	38.1	37.7	57.9	47.7
TinyImageNet	44.7	44.0	-	61.1

Table C.21: MAD Calibration Error (%)

	MSP	MCD	RotNet	TwinNet
CIFAR10	2.3	1.4	1.9	2.5
CIFAR100	9.3	3.5	9.1	27.4
TinyImageNet	12.0	4.3	-	28.9

Table C.22: Soft F1 Score (%)

	MSP	MCD	RotNet	TwinNet
CIFAR10	22.7	32.6	21.5	25.8
CIFAR100	38.7	49.2	38.7	44.8
TinyImageNet	50.0	56.7	-	65.0

Appendix D

Datasets

D.1 Class subsets

The following lists the class indices (from 0) for each randomised class subset in our benchmark. Each subset contains half of the total classes in the training dataset.

D.1.1 CIFAR-10

[8, 2, 5, 6, 3]

[7, 8, 2, 6, 4]

[5, 8, 7, 0, 4]

[3, 5, 6, 1, 4]

[3, 9, 0, 5, 4]

[2, 6, 1, 3, 7]

[6, 2, 0, 7, 8]

[7, 2, 5, 3, 4]

[7, 9, 0, 4, 2]

[1, 7, 9, 6, 8]

D.1.2 CIFAR-100

[19, 14, 43, 37, 66, 3, 79, 41, 38, 68, 2, 1, 60, 53, 95, 74, 92, 26, 59, 46, 90, 70, 50, 44, 76, 55, 21, 61, 6, 63, 42, 34, 84, 52, 35, 39, 45, 4, 5, 48, 32, 20, 83, 58, 47, 80, 17, 67, 81, 7]

[46, 49, 22, 58, 41, 98, 62, 29, 30, 51, 89, 78, 83, 79, 16, 38, 54, 6, 39, 93, 21, 36, 20, 61, 60, 96, 2, 11, 28, 9, 68, 52, 66, 97, 42, 69, 18, 85, 26, 99, 19, 65, 86, 47, 84, 77,

72, 8, 3, 59]

[17, 41, 92, 14, 68, 31, 89, 15, 21, 60, 12, 8, 39, 9, 7, 70, 58, 24, 86, 16, 83, 55, 26, 54, 19, 57, 46, 23, 36, 91, 81, 65, 84, 90, 88, 29, 38, 77, 40, 78, 20, 10, 28, 96, 95, 71, 73, 72, 42, 79]

[37, 62, 83, 14, 43, 9, 44, 31, 69, 57, 33, 87, 12, 91, 41, 23, 76, 29, 50, 68, 3, 4, 90, 72, 20, 59, 93, 96, 89, 47, 39, 27, 42, 13, 8, 88, 17, 84, 35, 95, 81, 0, 67, 55, 30, 36, 71, 61, 64, 10]

[24, 39, 35, 44, 55, 70, 82, 40, 91, 65, 2, 90, 18, 73, 97, 69, 52, 8, 29, 6, 34, 64, 38, 20, 50, 99, 7, 74, 54, 42, 23, 79, 58, 53, 47, 98, 33, 51, 36, 60, 62, 92, 16, 89, 48, 72, 49, 95, 14, 46]

[84, 36, 57, 51, 46, 78, 93, 14, 11, 59, 61, 38, 21, 90, 8, 25, 63, 9, 94, 97, 99, 3, 20, 55, 6, 81, 96, 30, 13, 16, 70, 69, 18, 58, 43, 87, 89, 64, 66, 33, 52, 48, 98, 92, 76, 79, 45, 67, 24, 54]

[7, 25, 71, 42, 47, 29, 63, 88, 50, 9, 48, 60, 26, 62, 98, 72, 91, 84, 33, 13, 96, 5, 90, 53, 67, 81, 39, 11, 10, 86, 45, 35, 38, 2, 94, 19, 36, 87, 27, 22, 51, 49, 23, 79, 55, 3, 73, 12, 75, 6]

[71, 28, 9, 4, 73, 34, 94, 92, 47, 37, 93, 76, 40, 70, 14, 16, 18, 80, 33, 78, 87, 60, 48, 82, 67, 81, 11, 25, 75, 53, 21, 95, 88, 66, 62, 35, 59, 29, 91, 36, 52, 55, 20, 12, 0, 58, 24, 69, 83, 46]

[53, 47, 43, 54, 33, 48, 0, 12, 44, 91, 2, 95, 76, 1, 74, 4, 87, 79, 6, 94, 23, 90, 97, 20, 40, 30, 81, 16, 52, 83, 45, 37, 80, 7, 82, 39, 98, 77, 78, 88, 57, 72, 27, 32, 71, 60, 38, 51, 26, 41]

[85, 72, 16, 18, 2, 38, 90, 30, 83, 61, 8, 44, 91, 13, 37, 46, 28, 58, 48, 76, 59, 14, 47, 81, 89, 70, 95, 33, 65, 6, 63, 62, 97, 43, 78, 1, 7, 24, 94, 64, 3, 21, 26, 42, 23, 87, 68, 99, 77, 41]

D.1.3 TinyImageNet

[59, 5, 20, 198, 52, 19, 162, 55, 69, 2, 98, 10, 75, 142, 124, 63, 109, 78, 111, 185, 154, 130, 61, 87, 102, 121, 136, 1, 47, 172, 159, 39, 76, 91, 35, 178, 127, 169, 46, 174, 190, 7, 26, 138, 58, 72, 103, 199, 56, 116, 24, 43, 101, 163, 21, 60, 175, 70, 90, 49, 119, 110, 95, 167, 193, 68, 165, 114, 67, 66, 120, 38, 196, 161, 99, 152, 83, 166, 117, 41, 80, 81, 32, 170, 48, 25, 53, 105, 17, 194, 51, 14, 82, 84, 184, 29, 3, 23, 147, 188]

[49, 191, 12, 172, 127, 40, 30, 170, 138, 57, 192, 72, 150, 146, 58, 26, 9, 115, 184, 198, 117, 36, 107, 113, 165, 161, 147, 65, 120, 21, 91, 188, 136, 71, 139, 2, 86, 116, 93,

106, 47, 142, 149, 152, 182, 78, 55, 25, 168, 194, 6, 95, 101, 151, 190, 173, 31, 74, 176, 144, 97, 20, 169, 137, 181, 11, 48, 183, 3, 59, 5, 13, 68, 0, 153, 108, 27, 89, 179, 100, 63, 141, 98, 61, 77, 43, 166, 185, 8, 60, 134, 121, 94, 34, 199, 196, 32, 80, 177, 180]

[174, 33, 173, 186, 22, 53, 134, 56, 143, 94, 55, 73, 130, 41, 14, 63, 16, 60, 111, 0, 2, 46, 86, 88, 61, 162, 77, 126, 198, 3, 135, 29, 25, 26, 93, 36, 83, 151, 139, 120, 163, 137, 30, 103, 98, 11, 54, 97, 101, 52, 42, 154, 123, 194, 150, 192, 112, 156, 9, 87, 165, 148, 95, 188, 32, 15, 122, 184, 69, 128, 66, 127, 51, 158, 181, 113, 172, 164, 118, 102, 12, 179, 197, 49, 64, 10, 108, 168, 185, 5, 121, 115, 149, 58, 6, 104, 99, 171, 21, 35]

[154, 0, 174, 59, 112, 170, 73, 7, 29, 32, 90, 96, 54, 134, 56, 81, 72, 187, 51, 100, 79, 171, 113, 118, 47, 52, 180, 39, 135, 105, 124, 19, 71, 99, 153, 189, 9, 142, 155, 156, 45, 195, 178, 104, 190, 183, 82, 166, 20, 177, 83, 131, 61, 77, 181, 48, 102, 11, 53, 2, 42, 1, 60, 193, 87, 117, 157, 122, 167, 66, 163, 149, 26, 147, 78, 35, 126, 13, 63, 5, 75, 70, 136, 55, 50, 141, 94, 95, 103, 132, 67, 194, 182, 197, 49, 172, 76, 140, 34, 89]

[0, 146, 158, 176, 197, 82, 75, 85, 80, 157, 71, 95, 138, 109, 147, 192, 126, 16, 87, 103, 177, 127, 11, 113, 94, 140, 170, 13, 69, 105, 81, 130, 160, 54, 37, 132, 64, 151, 51, 153, 9, 19, 199, 152, 3, 93, 48, 92, 73, 97, 120, 156, 188, 28, 145, 191, 178, 148, 142, 195, 77, 122, 162, 38, 42, 128, 115, 198, 155, 193, 88, 53, 90, 112, 124, 116, 76, 175, 29, 84, 5, 139, 196, 79, 86, 34, 102, 101, 70, 114, 98, 55, 118, 136, 137, 161, 194, 190, 144, 23]

[23, 182, 172, 21, 63, 11, 61, 32, 164, 7, 95, 173, 132, 120, 98, 138, 31, 16, 181, 85, 185, 166, 101, 129, 113, 51, 167, 88, 108, 136, 29, 183, 152, 93, 189, 139, 179, 55, 1, 5, 77, 47, 94, 14, 41, 125, 12, 131, 135, 24, 8, 40, 184, 127, 45, 142, 33, 149, 15, 195, 62, 6, 199, 124, 145, 39, 42, 90, 17, 50, 34, 66, 190, 37, 38, 86, 64, 140, 19, 171, 170, 81, 143, 126, 159, 187, 103, 78, 122, 156, 92, 26, 72, 83, 194, 10, 46, 36, 191, 30]

[179, 155, 23, 159, 96, 198, 42, 110, 128, 97, 95, 106, 65, 33, 102, 89, 132, 79, 104, 70, 173, 129, 30, 7, 168, 124, 157, 165, 101, 78, 87, 141, 105, 10, 64, 13, 90, 15, 195, 94, 125, 68, 69, 108, 131, 111, 92, 144, 12, 93, 137, 51, 18, 100, 32, 19, 17, 172, 192, 191, 38, 44, 148, 185, 46, 145, 115, 37, 170, 178, 80, 184, 21, 8, 103, 118, 162, 84, 14, 163, 177, 31, 27, 54, 45, 77, 167, 67, 164, 171, 194, 116, 56, 55, 143, 188, 63, 152, 39, 28]

[134, 91, 81, 108, 170, 39, 151, 47, 44, 186, 40, 74, 6, 25, 33, 128, 12, 109, 49, 88, 71, 129, 48, 4, 0, 9, 193, 155, 17, 122, 24, 162, 18, 149, 58, 184, 55, 130, 167, 118, 30, 65, 90, 199, 115, 112, 63, 3, 121, 57, 161, 56, 147, 135, 45, 34, 123, 191, 143, 166, 150, 189, 41, 62, 60, 160, 61, 43, 103, 165, 94, 96, 95, 195, 31, 126, 37, 27, 2, 77, 141, 172, 179, 46, 16, 163, 54, 148, 142, 192, 110, 87, 50, 119, 127, 132, 97, 23, 76, 125]

[40, 140, 33, 91, 13, 94, 148, 20, 196, 163, 31, 78, 16, 180, 37, 169, 72, 105, 152, 157, 117, 60, 62, 155, 190, 41, 59, 193, 132, 150, 1, 61, 34, 30, 26, 192, 8, 125, 64, 147, 154, 7, 83, 166, 58, 134, 14, 191, 2, 109, 142, 141, 129, 182, 183, 144, 120, 161, 96, 92, 143, 103, 36, 115, 27, 82, 75, 95, 23, 89, 135, 24, 0, 111, 199, 197, 18, 171, 189, 121, 68, 151, 56, 54, 173, 179, 43, 5, 149, 6, 80, 65, 51, 38, 79, 172, 42, 119, 127, 99]

[114, 189, 198, 151, 5, 69, 24, 15, 51, 112, 93, 155, 6, 175, 122, 47, 44, 64, 157, 149, 106, 10, 16, 126, 40, 11, 177, 23, 140, 60, 116, 18, 77, 73, 117, 31, 54, 100, 70, 130, 172, 163, 14, 58, 20, 90, 7, 145, 46, 89, 180, 110, 111, 188, 98, 161, 35, 3, 129, 94, 181, 144, 142, 53, 66, 108, 105, 2, 170, 196, 160, 22, 39, 164, 36, 118, 91, 42, 29, 82, 146, 96, 143, 182, 19, 165, 87, 167, 168, 179, 183, 34, 152, 158, 136, 8, 38, 28, 49, 26]