No True State-of-the-Art? OOD Detection Methods are Inconsistent across Datasets

Fahim Tajwar [1] **Ananya Kumar** [*1] **Sang Michael Xie** [*1] **Percy Liang** [1]

Abstract

Out-of-distribution detection is an important component of reliable ML systems. Prior literature has proposed various methods (e.g., MSP (Hendrycks & Gimpel, 2017), ODIN (Liang et al., 2018), Mahalanobis (Lee et al., 2018)), claiming they are state-of-the-art by showing they consistently outperform previous methods on a selected set of in-distribution (ID) and out-of-distribution (OOD) datasets. In this work, we show that none of these methods are inherently better at OOD detection than others on a standardized set of 16 (ID, OOD) pairs. We give possible explanations for these inconsistencies with simple toy datasets where whether one method outperforms another depends on the structure of the ID and OOD datasets in question. Finally, we show that a method outperforming another on a certain (ID, OOD) pair may not do so in a low data regime. In the low-data regime, we propose a distance-based method, Pairwise OOD detection (POD), which is based on Siamese networks and improves over Mahalanobis by sidestepping the expensive covariance estimation step. Our results suggest that the OOD detection problem may be too broad, and we should consider more specific structures for leverage.

1. Introduction

When the test distribution matches the training distribution, deep neural networks (DNN) generalize well (Krizhevsky et al., 2012; Simonyan & Zisserman, 2015; Cho et al., 2014; Zhang et al., 2017). However, when deploying a neural network in the real world, this assumption often does not hold. For example, a neural network trained on a certain set of diseases may face an unseen disease during test time, and ideally should flag such an occurrence for deferral to a human expert (Mozannar & Sontag, 2020; Rajpurkar et al., 2020). This work focuses on OOD detection (Hendrycks & Gimpel, 2017; Liang et al., 2018; Lee et al., 2018), the binary classification task of detecting whether a test example is sampled from a *known* distribution $\mathcal{D}_{in}$ (an in-distribution (ID) example) or an *unknown* distribution $\mathcal{D}_{out}$ (an out-of-distribution (OOD) example).

	MSP		ODIN		Mahalanobis		POD	
	Full	Low	Full	Low	Full	Low	Full	Low
MSP	—	—	2/16	3/16	2/16	9/16	3/16	3/16
ODIN	14/16	13/16	—	—	4/16	13/16	4/16	13/16
Mahalanobis	14/16	7/16	12/16	3/16	—	—	12/16	6/16
POD	13/16	13/16	12/16	3/16	4/16	10/16	—	—

Table 1. Comparison table showing the number of ID-OOD dataset pairs (out of 16) on which the row method outperforms the column method on OOD detection AUROC, without using any OOD samples for training/hyper-parameter tuning. No method does consistently better than any other method across the 16 pairs (none of the numbers are 0 or 16). Comparisons are also sensitive to in-distribution training dataset sizes — for example, Mahalanobis (third row) does better than ODIN on 12/16 pairs using the full ID train set, but on only 3/16 pairs in the low data regime.

In this work, we consider two popular groups of OOD detection methods: predictive score-based (e.g., MSP (Hendrycks & Gimpel, 2017), ODIN (Liang et al., 2018)) and distance-based (e.g., Mahalanobis (Lee et al., 2018)) methods. Prior literature shows empirically that their methods do better than other commonly used baselines on a number of ($\mathcal{D}_{in}$, $\mathcal{D}_{out}$) pairs. When we standardize training details and test on a larger number of datasets, we find that no single method consistently outperforms others across different ($\mathcal{D}_{in}$, $\mathcal{D}_{out}$) pairs on the OOD detection task.

Specifically, under the assumption that no OOD data is seen during training/hyper-parameter tuning (similar to Hsu et al. (2020)), we run experiments on a set of 3 ID datasets (CIFAR-10, CIFAR-100 and SVHN) and 7 OOD datasets (CIFAR-10, CIFAR-100, SVHN, CelebA, STL-10, TinyImageNet, LSUN) with a standardized training procedure and show that none of three popular OOD detection methods (MSP, ODIN, Mahalanobis) consistently does better than

[*]Equal contribution [1]Department of Computer Science, Stanford University, Stanford, California, USA. Correspondence to: Fahim Tajwar <tajwar@cs.stanford.edu>.

Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning., Copyright 2021 by the author(s).

the others (see Table 1) on the 16 different ($\mathcal{D}_{in}$, $\mathcal{D}_{out}$) pairs in terms of OOD detection AUROC. Comparisons are also sensitive to in-distribution training dataset sizes — for example, Mahalanobis does better than ODIN on 12 out of 16 cases with the entire ID train dataset; but on only 3 out of 16 cases when 10% of the ID train data is used, so there is no clear winner even if we average the metrics across datasets.

To examine possible reasons why these inconsistencies can arise, and show that these issues are fundamental and not specific to implementation details, we provide toy datasets that demonstrate whether predictive score-based methods do better or worse than distance-based methods depends on the particular ($\mathcal{D}_{in}$, $\mathcal{D}_{out}$) pair, and neither type of method is universally better. Specifically, both a linear classifier and a distance based classifier achieve 100% ID test classification accuracy in the two toy datasets in $\mathbb{R}^2$, but —

- On toy dataset 1 (Figure 1), a distance-based method achieves 100% AUROC on the OOD detection task, whereas MSP (based on linear classifier) achieves 0%.
- In contrast, on toy dataset 2 (Figure 2), a distance-based method achieves 32.5% AUROC in contrast to MSP's 100% AUROC on the OOD detection task.

Given that inconsistencies are fundamental in the broad OOD detection setting, we consider more specific settings where differences between methods may be more consistent. For example, with access to some OOD data for hyper-parameter tuning, we see more consistent trends for OOD detection methods. In the low-data regime, we propose a distance-based method, Pairwise OOD detection (POD), that is based on Siamese networks (Koch et al., 2015) and improves the Mahalanobis method (Lee et al., 2018) by crucially sidestepping the expensive covariance matrix estimation step. POD improves over MSP and Mahalanobis in 13 and 10 out of 16 dataset pairs respectively in the low data regime (Table 1).

Our contributions can be summarized as:

1. We run three popular baselines (MSP, ODIN, Mahalanobis) and POD on a standard set of 16 ($\mathcal{D}_{in}$, $\mathcal{D}_{out}$) pairs, and show that no method performs better than the others consistently on the full data setting.
2. We present toy datasets in $\mathbb{R}^2$ and show that none of the two popular classes of OOD detection methods (predictive score- and distance-based methods) do better than the other consistently, and their performance depends on the particular ($\mathcal{D}_{in}$, $\mathcal{D}_{out}$) pair.
3. Finally, we show that this inconsistency is even more pronounced in the low-data setting, i.e., methods that tend to do better than another in the high data setting may do worse when the ID training dataset size has decreased, so even if we average metrics across datasets there is no clear best method.

Given the inconsistencies in the broad OOD detection problem setting, we believe more specific settings (e.g., some access to OOD samples for hyper-parameter tuning, low data regime) are necessary for targeted improvements.

2. Problem Description

Here we formally define the OOD detection problem following prior works such as Liang et al. (2018). Let $\mathcal{X}$ be the input space, $\mathcal{Y}$ be the label space, and $\mathcal{D}_{in}$ be a known distribution over $\mathcal{X} \times \mathcal{Y}$ with C classes. We have access to a set of N examples, $D_{in}^{train} = \{(x_i, y_i)\}_{i=1}^{N}$ sampled from $\mathcal{D}_{in}$ for training. Consider any $x \in \mathcal{X}$, a model trained on D_{in}^{train} will output a score $S(x) \in \mathbb{R}$, where $S(x)$ should be higher for in-distribution x. Then we can use $S(x)$ as a heuristic for OOD detection, i.e., we pick a threshold T, and say x is ID if $S(x) \geq T$, or otherwise x is OOD.

During *test time*, we have examples D_{out}^{test} drawn from an unseen distribution $\mathcal{D}_{out}$. We also have a set of examples drawn from $\mathcal{D}_{in}$ that are not in D_{in}^{train}, we denote this as D_{in}^{test}. Then OOD detection becomes a binary classification problem where examples from D_{in}^{test} are ID or positive, and examples from D_{out}^{test} are OOD or negative. We use two popular metrics (Hendrycks et al., 2019) to measure the performance of an OOD detection method:

1. **AUROC**: Area under the receiver operating characteristic curve, which is the plot of true positive rate vs false positive rate for a binary classification problem.
2. **FNR@95**: False negative rate (fraction of ID examples misclassified as OOD) when 95% of OOD examples are classified as OOD (true negative). Note that Hendrycks et al. (2019) denotes this as FPR@95, since they denote OOD examples as positive instead of negative and vice versa.

3. Methods

We compare four OOD detection methods (3 popular baselines and our method) in this work.

1. **MSP** (Hendrycks & Gimpel, 2017): Maximum softmax probability (MSP) is a commonly used predictive score based method. We first train a neural network f to classify the C in-distribution classes. Given input x, let $f^i(x)$ denote the model's confidence that x has label i. The temperature-scaled softmax score for class

i is:

$$S^i_{soft}(x;T) = \frac{\exp\left(f^i(x)/T\right)}{\sum_{j=1}^{C}\exp\left(f^j(x)/T\right)} \quad (1)$$

The MSP score for x is simply the maximum probability across the classes:

$$S_{MSP}(x) = \max_{i\in\{1,\ldots,C\}} S^i_{soft}(x;T=1) \quad (2)$$

2. **ODIN** (Liang et al., 2018): ODIN builds on MSP by temperature scaling with some $T > 0$ and pre-processes the inputs with noise $\epsilon > 0$ to improve OOD detection performance. Let $\hat{x} = G_{ODIN}(x;\epsilon,T)$ be the pre-processed version of $x \in \mathcal{X}$ (Appendix B.1). Then the ODIN score for x is

$$S_{ODIN}(x;T) = \max_{i\in\{1,\ldots,C\}} S^i_{soft}(\hat{x};T) \quad (3)$$

3. **Mahalanobis** (Lee et al., 2018): Mahalanobis is a popular distance based method. Let $h(x)$, μ_i and Σ be the neural network representation, class mean for label i, and covariance matrix at the neural network's penultimate layer respectively. Let $\hat{x} = G_{Maha}(x;\epsilon)$ be the pre-processed version of input x. For details on how μ_i, Σ and $\hat{x}$ are calculated, see Appendix B.2. We define

$$S_{Maha}(x) = \max_{i=\{1,\ldots,C\}} \{-(h(\hat{x})-\mu_i)^T\Sigma^{-1} (h(\hat{x})-\mu_i)\} \quad (4)$$

Note that Lee et al. (2018) use the weighted sum of the Mahalanobis distance from each layer of the neural network. However, the weights are tuned using OOD samples. Following more recent work (Liu et al., 2020), we use the Mahalanobis distance in only the penultimate layer.

4. **Pairwise OOD Detection (POD)** (Ours): We propose a distance-based method for the low-data regime – "Pairwise OOD Detection" (POD). Let $h(x) \in \mathbb{R}^d$ be the neural network's penultimate layer's representation for test sample x. Suppose we have M images from each of the C ID classes i.e., $\{\{p_{ij}\}_{j=1}^{M}\}_{i=1}^{C}$ where p_{ij} is the j-th example from the i-th ID class. We define

$$S^i_{POD}(x) = \frac{1}{M}\sum_{j=1}^{M}||h(x)-h(p_{ij})||_2^2 \quad (5)$$

$$S_{POD}(x) = -\min_{i\in\{1,\cdots,C\}} S^i_{POD}(x) \quad (6)$$

Instead of this particular setup, we could take the min or average over all distances. In Appendix D.1, we provide justification for our choice.

Among these methods, ODIN and Mahalanobis require extra hyper-parameters to be tuned using OOD samples. We do not typically have access to $\mathcal{D}_{out}$, and methods such as Outlier exposure (Hendrycks et al., 2019) or Energy based fine-tuning (Liu et al., 2020) do better when there is some access to OOD samples. We record results with standardized hyper-parameters in Table 1, and OOD sample tuned hyper-parameters in Table 4.4. Please see Appendix B for details on OOD sample dependent hyper-parameters.

4. OOD Detection methods are inconsistent

4.1. Toy datasets

Using toy datasets, we demonstrate that an OOD detection method's relative performance depends on the $(\mathcal{D}_{in}, \mathcal{D}_{out})$ pair in question. Specifically, we present two toy datasets in $\mathbb{R}^2$ (see Appendix C on how to generate them). For the predictive score based model, we consider MSP on top of a linear classifier trained on D_{in}^{train} dataset. For the distance based method, we use the negative of the minimum distance from the test sample x to any sample in D_{in}^{train} for ID classification or OOD detection task.

Now on both of these datasets, the predictive score and distance-based methods achieve 100% accuracy on D_{in}^{test}. However, on toy dataset 1 (Figure 1), the OOD examples are far away from the decision boundary and any ID examples. This makes the MSP predict higher confidence scores on the OOD examples than the ID examples, and so MSP achieves 0% AUROC on the OOD detection task. However, for the same reason the distance-based method achieves 100% AUROC.

In contrast, on toy dataset 2 (Figure 2), the two ID classes are linearly separable and the majority of the ID samples are spread out. The OOD examples are closer to the decision boundary than ID examples, and there are ID outliers very close to the OOD examples as well. Due to OOD examples being closer to the decision boundary, MSP achieves 100% AUROC. However, the distance based method is fooled by the ID outliers, and achieves only 32.5% AUROC.

4.2. Standard/Full data OOD detection setting

We demonstrate the inconsistency among popular OOD detection methods by designing a standardized set of experiments (Appendix A). We use the full ID train dataset, and have no access to any OOD sample for training/tuning.

Datasets: We use 3 ID datasets (CIFAR-10, CIFAR-100 and SVHN) and 7 OOD datasets (CIFAR-10, CIFAR-100, SVHN, CelebA, STL-10, TinyImageNet and LSUN) to form 16 different $(\mathcal{D}_{in}, \mathcal{D}_{out})$ pairs. See Appendix A.3 for (ID, OOD) pairs that are excluded because of class overlaps.

Model Architecture: We use ResNet-34 (He et al., 2016)

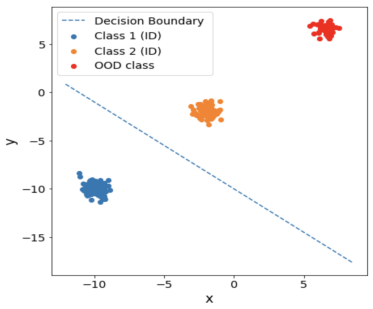

Figure 1. Toy Dataset 1. Here, distance-based methods do better than MSP due to the OOD examples being far away from the decision boundary between classes, and thus fooling MSP into giving them a higher confidence than ID samples.

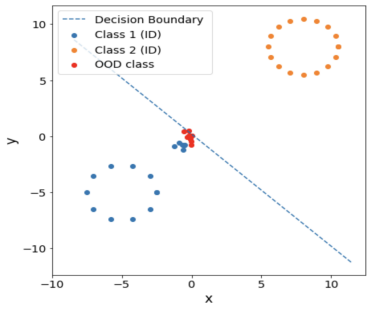

Figure 2. Toy Dataset 2. Here, MSP does better than distance based methods since OOD examples are very close to the decision boundary between classes and so have lower confidence scores, whereas ID outliers close to the OOD examples fool the distance based method.

as the network backbone for all of our experiments.

The inconsistency across different methods are presented in Table 1.

4.3. Low data OOD detection setting

To investigate whether there is more consistency in specific settings, we do the same experiments in a low-data regime, where we use only 10% of each ID train dataset $\mathcal{D}_{in}^{train}$. In the low-data regime (Table 1), ODIN does better than each of the other methods on $13/16$ (ID, OOD) pairs, whereas in the full-data regime ODIN only did better than Mahanobis and POD on $4/16$ pairs. In contrast, Mahalanobis performs much worse in the low data regime, which we attribute to the statistically expensive covariance matrix estimation step. Finally, POD outperforms Mahalanobis in the low data setting, possibly by not requiring covariance estimation. This reveals that methods' performance depends on not only the (ID, OOD) pair but also the ID dataset size.

4.4. With Access to OOD Samples

We further specialize the setting by doing the experiments in Sections 4.2 and 4.3 while letting ODIN and Mahalanobis use OOD samples to tune hyper-parameters (Appendix B). Table 4.4 shows that ODIN and Mahalanobis improve in the full data setting when using OOD examples. However, significant inconsistency remains between ODIN and Mahalanobis themselves, and specially in the full vs low data regime, where their performance order switches.

5. Discussion & Conclusion

Our fundamental result is that the current setting of OOD detection is too broad. Some prior works have also discussed this issue and proposed different settings. For example, the ID and OOD distributions can be very different (e.g., CIFAR-10 vs SVHN) in our setting. Ahmed & Courville (2020) argue that only the case when OOD examples are semantically similar to ID ones is of practical importance, and propose benchmarks to withhold one ID class during training and detecting examples from the held out class during testing as OOD. Ruff et al. (2020) considers detecting low probability examples (e.g., blurred images, objects with defects) from the training distribution instead of OOD examples from an unseen distribution and find a lack of consistency among these in-distribution anomaly detection methods. Some failure modes of earlier methods have also been discovered — Hsu et al. (2020) constructs a dataset containing semantic and non-semantic shifts based on DomainNet (Peng et al., 2019) and shows that Mahalanobis method performs worse than MSP baseline, Ren et al. (2021) shows the same for semantically similar distributions (CIFAR-10 vs CIFAR-100) and Huang & Li (2021) for large semantic spaces (e.g., ImageNet). We suggest further studies into the failure modes and possible correlation between dataset statistics and the rank of performance of an OOD detection method, and conclude that the community might benefit from moving away from the practice of coming up with a method and testing it on certain (ID, OOD) pairs, and look for more specific problem settings.

	MSP		ODIN		Mahalanobis		POD	
	Full	Low	Full	Low	Full	Low	Full	Low
MSP	—	—	2/16	0/16	2/16	6/16	3/16	3/16
ODIN	14/16	16/16	—	—	5/16	10/16	8/16	16/16
Mahalanobis	14/16	10/16	11/16	6/16	—	—	13/16	10/16
POD	13/16	13/16	8/16	0/16	3/16	6/16	—	—

Table 2. Comparison table showing how many ID-OOD dataset pairs (out of 16) the row method outperforms the column method on OOD detection AUROC. In contrast to Table 1, we let ODIN and Mahalanobis use OOD samples for hyper-parameter tuning in this table.

Acknowledgements

We thank Chelsea Finn, Tengyu Ma, Yining Chen, Tom Knowles, and Dan Hendrycks for giving us comments and suggestions during the course of this work, Weitang Liu for his clarifications on CelebA dataset transformations (Liu et al., 2020), and anonymous reviewers for their suggestions on how to improve the paper. We would also like to thank Sumaita Sadia Rahman for her help with proof-reading the entire paper. This work was supported by NSF Award Grant No. 1805310. Fahim Tajwar was supported by Stanford VPUE (Vice Provost for Undergraduate Education) via the Computer Science department's CURIS program, Ananya Kumar was supported by a Stanford Graduate Fellowship, and Sang Michael Xie was supported by an NDSEG fellowshop.

Reproducibility

We have deposited all our code and instructions on how to replicate the results in this paper in this GitHub repository.

References

Ahmed, F. and Courville, A. Detecting semantic anomalies. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3154–3162, Apr. 2020. doi: 10.1609/aaai.v34i04.5712. URL https://ojs.aaai.org/index.php/AAAI/article/view/5712.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety. In *arXiv*, 2016. URL https://arxiv.org/abs/1606.06565.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/chen20j.html.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179.

Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In Gordon, G., Dunson, D., and Dudík, M. (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pp. 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL http://proceedings.mlr.press/v15/coates11a.html.

Deng, J., Socher, R., Fei-Fei, L., Dong, W., Li, K., and Li, L.-J. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 00, pp. 248–255, 06 2009. doi: 10.1109/CVPR.2009.5206848. URL https://ieeexplore.ieee.org/abstract/document/5206848/.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6572.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distributio examples in neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Hkg4TI9xl.

Hendrycks, D., Mazeika, M., and Dietterich, T. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyxCxhRcY7.

Høye, T. T., Ärje, J., Bjerge, K., Hansen, O. L. P., Iosifidis, A., Leese, F., Mann, H. M. R., Meissner, K., Melvad, C., and Raitoharju, J. Deep learning and computer vision will transform entomology. *Proceedings of the National Academy of Sciences*, 118(2), 2021. ISSN 0027-8424. doi: 10.1073/pnas.2002545117. URL https://www.pnas.org/content/118/2/e2002545117.

Hsu, Y.-C., Shen, Y., Jin, H., and Kira, Z. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Huang, R. and Li, Y. Mos: Towards scaling out-of-distribution detection for large semantic space. *arXiv preprint arXiv:2105.01879*, 2021. URL https://arxiv.org/abs/2105.01879.

Koch, G., Zemel, R., and Salakhutdinov, R. Siamese neural networks for one-shot image recognition, 2015. URL https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf.

Kolesnikov, A., Zhai, X., and Beyer, L. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Krizhevsky, A. Learning multiple layers of features from tiny images. *MSc thesis, Department of Computer Science, University of Toronto*, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

LeCun, Y. and Cortes, C. http://yann.lecun.com/exdb/mnist/, 2010. URL http://yann.lecun.com/exdb/mnist/.

Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/abdeb6f575ac5c6676b747bca8d09cc2-Paper.pdf.

Liang, S., Li, Y., and Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1VGkIxRZ.

Liu, W., Wang, X., Owens, J., and Li, Y. Energy-based out-of-distribution detection. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21464–21475. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/f5496252609c43eb8a3d147ab9b9c006-Paper.pdf.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Loshchilov, I. and Hutter, F. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Mozannar, H. and Sontag, D. Consistent estimators for learning to defer to an expert. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7076–7087. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/mozannar20b.html.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Nguyen, A., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015.

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Prabhu, V., Kannan, A., Ravuri, M., Chablani, M., Sontag, D. A., and Amatriain, X. Prototypical clustering networks for dermatological disease diagnosis. *CoRR*, abs/1811.03066, 2018. URL http://arxiv.org/abs/1811.03066.

Prabhu, V., Kannan, A., Ravuri, M., Chaplain, M., Sontag, D. A., and Amatriain, X. Few-shot learning for dermatological disease diagnosis. In Doshi-Velez, F., Fackler, J., Jung, K., Kale, D. C., Ranganath, R., Wallace, B. C., and Wiens, J. (eds.), *Proceedings of the Machine Learning for Healthcare Conference, MLHC 2019, 9-10 August 2019, Ann Arbor, Michigan, USA*, volume 106 of *Proceedings of Machine Learning Research*, pp. 532–552. PMLR, 2019a. URL http://proceedings.mlr.press/v106/prabhu19a.html.

Prabhu, V., Kannan, A., Tso, G. J., Katariya, N., Chablani, M., Sontag, D. A., and Amatriain, X. Open set medical diagnosis. *CoRR*, abs/1910.02830, 2019b. URL http://arxiv.org/abs/1910.02830.

Rajpurkar, P., Joshi, A., Pareek, A., Chen, P., Kiani, A., Irvin, J., Ng, A. Y., and Lungren, M. P. Chexpedition: Investigating generalization challenges for translation of chest x-ray algorithms to the clinical setting. In *arXiv*, 2020. URL https://arxiv.org/abs/2002.11379.

Reed, C. J., Metzger, S., Srinivas, A., Darrell, T., and Keutzer, K. Selfaugment: Automatic augmentation policies for self-supervised learning. In *arXiv*, 2021. URL https://arxiv.org/abs/2009.07724.

Ren, J., Fort, S., Liu, J., Roy, A. G., Padhy, S., and Lakshminarayanan, B. A simple fix to mahalanobis distance for improving near-ood detection. In *arXiv*, 2021. URL https://arxiv.org/abs/2106.09022.

Ruff, L., Kauffmann, J. R., Vandermeulen, R. A., Montavon, G., Samek, W., Kloft, M., Dietterich, T. G., and Müller, K. A unifying review of deep and shallow anomaly detection. In *arXiv*, 2020. URL https://arxiv.org/abs/2009.11732.

Shi, S., Malhi, I., Tran, K., Ng, A. Y., and Rajpurkar, P. Chexseen: Unseen disease detection for deep learning interpretation of chest x-rays. In *arXiv*, 2021. URL https://arxiv.org/abs/2103.04590.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.1556.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6199.

Yu, F., Zhang, Y., Song, S., Seff, A., and Xiao, J. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. URL https://arxiv.org/abs/1506.03365.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In Richard C. Wilson, E. R. H. and Smith, W. A. P. (eds.), *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 87.1–87.12. BMVA Press, September 2016. ISBN 1-901725-59-6. doi: 10.5244/C.30.87. URL https://dx.doi.org/10.5244/C.30.87.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Sy8gdB9xx.

A. Experiment setup

A.1. In-distribution (ID) datasets

We use three commonly used In-distribution datasets in our experiments:

1. **CIFAR-10** (Krizhevsky, 2009): CIFAR-10 contains 50,000 train and 10,000 test images, separated into 10 disjoint classes. All images are of size 32 x 32, and have three channels. The image classes are similar but disjoint from those of CIFAR-100.

2. **CIFAR-100** (Krizhevsky, 2009): CIFAR-100 contains 50,000 train and 10,000 test images of size 32 x 32, similar to CIFAR-10. However, the dataset has 100 classes, which can be grouped into 20 super-classes. Each super-class in CIFAR-100 containing 5 finer-grained classes. The classes in CIFAR-10 and CIFAR-100 are mutually disjoint.

3. **SVHN** (Netzer et al., 2011): SVHN contains 73,257 train and 26032 test images, each with size 32 x 32. The original dataset contains some extra train examples which we do not use in our experiments. There are 10 classes in this dataset representing the 10 different digits in English.

For all ID datasets, we use the standard train-test split, and refer to them as $\mathcal{D}_{in}^{train}$ and $\mathcal{D}_{in}^{test}$, respectively.

A.2. Out-of-distribution (OOD) datasets

We use all of the three (CIFAR-10, CIFAR-100 and SVHN) datasets mentioned in A.1 as OOD datasets. Furthermore, we use the following additional OOD datasets:

1. **CelebA** (Liu et al., 2015): This dataset contains images of celebrities' faces. We resize the CelebA images into 32 x 32, and use the standard test split, which contains 19,962 images.

2. **STL-10** (Coates et al., 2011): This dataset contains almost the same classes as CIFAR-10, but the image sizes are larger. We resize the STL-10 images into 32 x 32, and use the standard test split, which contains 8000 images.

3. **TinyImageNet** (Deng et al., 2009; Liang et al., 2018): TinyImageNet contains a subset of the ImageNet dataset (Deng et al., 2009) which has 10,000 test images divided into 200 different classes. Liang et al. (2018) further describes two datasets created by randomly cropping and resizing the test images to size 32 x 32 respectively. We use the resized dataset in our work here. Descriptions of this dataset (with links to download it as well) are given here: https://github.com/ShiyuLiang/odin-pytorch

4. **LSUN** (Yu et al., 2015; Liang et al., 2018): The LSUN dataset, also known as the **L**arge-scale **S**cene **UN**derstanding dataset, has 10,000 test images divided into 10 different classes. Similar to the TinyImageNet dataset, Liang et al. (2018) constructs two datasets by randomly cropping and resizing the test images into size 32 x 32. We use the resized dataset in our work here. Further details, as well as intruction on how to download the dataset, can be obtained here: https://github.com/ShiyuLiang/odin-pytorch

For all OOD datasets where there is a standard train-test split (CIFAR-10, CIFAR-100, SVHN, CelebA, STL-10) we use only the test split as OOD samples, and refer to it as $\mathcal{D}_{out}^{test}$. For the datasets where such a split is not available (TinyImageNet and LSUN), we use the entire dataset as $\mathcal{D}_{out}^{test}$.

A.3. Constructing the set of (ID, OOD) pairs

We construct 16 total (ID, OOD) pairs from the 3 ID datasets in A.1 and 7 OOD datasets in A.2. One may notice that we can form 21 different (ID, OOD) pairs; however, (CIFAR-10, CIFAR-10), (CIFAR-100, CIFAR-100) and (SVHN, SVHN) are not valid pairs.

Furthermore, we do not consider ($\mathcal{D}_{in}$, $\mathcal{D}_{out}$) pairs that share significant number of common classes and were not used in earlier literature, for example (CIFAR-10, STL-10) and (CIFAR-100, CelebA). CIFAR-10 and STL-10 share 9 out of 10 classes among them, and CIFAR-100 contains a "people" super-class containing facial images similar to CelebA. Hence we exclude (CIFAR-10, STL-10) and (CIFAR-100, CelebA), and arrive at 16 (ID, OOD) pairs for our experiments. Note that (CIFAR-10, TinyImageNet) and (CIFAR-100, TinyImageNet) may have some pretty similar classes; however the number of classes shared is pretty low and earlier literature (Lee et al., 2018; Liang et al., 2018) uses these pairs, so we decided to use them in our work as well.

A.4. Model architecture and training details

We use a ResNet (He et al., 2016) architecture with 34 layers for all our experiments (unless explicitly mentioned otherwise), similar to the works of Lee et al. (2018) and Hsu et al. (2020). All neural networks are trained using SGD with Nesterov momentum, and with a cosine learning rate scheduler (Loshchilov & Hutter, 2017).

We standardize the training details for each network/method, i.e., the number of epochs and the number of images a neural

Training Dataset	Percentage of Training Data Used	Method	Training Detail			
			# Epochs	Batch size	Initial Learning Rate	# Image pairs per epoch
CIFAR-10 & CIFAR-100	100%	MSP / ODIN / Mahalanobis / POD	200	128	0.1	N/A
		POD + Fine-tune	(Classifier) 175 (FT) 25	128 32	0.1 0.01	N/A 25,000
	10%	MSP / ODIN / Mahalanobis / POD	1000	128	0.1	N/A
		POD + Fine-tune	(Classifier) 750 (FT) 250	128 32	0.1 0.01	N/A 2,500
SVHN	100%	MSP / ODIN / Mahalanobis / POD	200	128	0.1	N/A
		POD + Fine-tune	(Classifier) 175 (FT) 25	128 32	0.1 0.01	N/A 36,000
SVHN	10%	MSP / ODIN / Mahalanobis / POD	1000	128	0.1	N/A
		POD + Fine-tune	(Classifier) 750 (FT) 250	128 32	0.1 0.01	N/A 3,600

Table 3. Training details for the neural networks we use. Note that the same neural network is used for MSP, ODIN, Mahalanobis and POD methods. For POD + Fine-tune, we first train a classifier for a fewer number of epochs, and then fine-tune it for some additional number of epochs on the pairwise classification task. For fine-tuning, a network sees N same class pairs, and N different class pairs - a total 2N pairs of images. We choose the number of image pairs the network sees during each epoch such that the total number of images it sees does not exceed the number of images a classifier sees during its training, to make sure POD + Fine-tune does not get any "extra" training benefit over the other methods.

net sees during each epoch, so as to ensure that no method gains an unfair advantage above the other. The training detail for each method is recorded in Table 3. For more details on the POD + Fine-tune method (e.g., its training algorithm), please see Appendix E.1.

A.5. Per class samples used for POD methods

As discussed in section 3, we need M images per each of the C classes in the In-distribution dataset for POD. Note that all of these images belong to $\mathcal{D}_{in}^{train}$ and we do not use any OOD dataset.

For all of the experiments in this work, we choose $M = 20$. We have experimented with different values of M, and saw that using lower values of M usually harms the OOD detection performance. However, choosing $M > 20$ does not increase the performance substantially, and only increases the computational cost.

A.6. Low data experiment details

For our low data experiments, we use 10% of the ID train datasets ($\mathcal{D}_{in}^{train}$) for the purpose of training and testing. For the sake of reproducibility and consistency, we form the low data train dataset ($\mathcal{D}_{in}^{train,low}$) as follows: for each of the C classes in the standard datasets, we take the first 10% of the examples, instead of a random 10% sample, (i.e., for CIFAR-10, we take the first 500 samples from each of the ID classes from the training dataset) and include them to form the new training dataset. We have experimented with choosing the low data subset randomly, it does not add any significant variation in the reported numbers.

B. Discussion of OOD sample dependent hyper-parameters for ODIN / Mahalanobis

B.1. Input pre-processing for ODIN

In addition to temperature scaling the softmax score by some $T > 0$ (Equation 1), ODIN (Liang et al., 2018) uses input pre-processing by some noise $\epsilon > 0$ to improve OOD detection. Consider input $x \in \mathcal{X}$. We define $S_{ODIN,w/o}(x)$

Method	Hyper-parameters	Search Grid	Fixed Value
ODIN	Temperature (T)	1, 10, 100, 1000	1000
	Noise (ϵ)	0, 0.0005, 0.001, 0.0014, 0.002, 0.0024, 0.005, 0.01, 0.05, 0.1, 0.2	0
Mahalanobis	Noise (ϵ)	0, 0.0005, 0.001, 0.0014, 0.002, 0.0024, 0.005, 0.01, 0.05, 0.1, 0.2	0

Table 4. Hyper-parameter search grid for ODIN & Mahalanobis. For each (ID, OOD) pair, we evaluate the method on all pairs of (T, ϵ) from this table, and record the best results in Table 4.4. When we do not have access to OOD samples, we use the fixed values for the hyper-parameters, and those results are recorded in Table 1.

to the ODIN score for input x without any pre-processing:

$$S_{ODIN,w/o}(x;T) = \max_{i\in\{1,\cdots,C\}} S^i_{soft}(x;T) \tag{7}$$

where $S^i_{soft}(x;T)$ is defined in Equation (1). Let $\hat{x} = G_{ODIN}(x;\epsilon,T)$ to be the pre-processed version of x. We have,

$$G_{ODIN}(x;\epsilon,T) = x - \epsilon\,\text{sign}(-\nabla_x \log S_{ODIN,w/o}(x;T)) \tag{8}$$

B.2. Class means, covariance matrix and input pre-processing for Mahalanobis

Note that in the original experiments from Lee et al. (2018), one uses the weighted sum of the Mahalanobis distances from all residual blocks of a ResNet (He et al., 2016). In our own work, we follow more recent literature (e.g., Liu et al. (2020)) and use only the Mahalanobis distance of the neural network's penultimate (second-to-last) residual block's representation. The reason for this is simple — Lee et al. (2018) assumes access to some OOD examples in order to tune the weights of individual layers. They mention that the layers that do not contribute to OOD detection have their respective weights set close to zero, and in practice become "ineffective" layers. However, in our setting, $\mathcal{D}_{out}$ is unseen, and no OOD examples are known for training or hyper-parameter tuning, so the weights cannot be properly chosen. Hence we use only the penultimate layer.

Let $h(x)$, μ_i and Σ be the neural network representation, class mean for label i, and covariance matrix at the neural network's penultimate layer, respectively. μ_i and Σ are defined, using the training examples $\{(x_j, y_j)\}_{j=1}^N$ sampled from $\mathcal{D}_{in}$, as follows:

$$\mu_i = \frac{1}{N_i} \sum_{j:y_j=i} h(x_j) \tag{9}$$

$$\Sigma = \frac{1}{N} \sum_{i=1}^{C} \sum_{j:y_j=i} (h(x_j) - \mu_i)(h(x_j) - \mu_i)^T \tag{10}$$

where N_i and N are the number of training examples in class i and the total number of training examples, respectively.

Similar to ODIN, Mahalanobis (Lee et al., 2018) also employs input pre-processing by noise $\epsilon > 0$. Consider some $x \in \mathcal{X}$. We first find the closest class c to example x in the penultimate layer, i.e.,

$$c = \underset{i=1}{\overset{C}{\text{argmin}}} \{(h(x) - \mu_i)^T \Sigma^{-1} (h(x) - \mu_i)\} \tag{11}$$

Let $\hat{x} = G_{Maha}(x;\epsilon)$ be the pre-processed version of input x. We define it as follows:

$$G_{Maha}(x;\epsilon) = x - \epsilon\,\text{sign}(\nabla_x (h(x) - \mu_i)^T \Sigma^{-1} (h(\hat{x}) - \mu_i)) \tag{12}$$

B.3. OOD sample dependent hyper-parameters for ODIN

The hyper-parameters T and ϵ for ODIN require OOD samples to be tuned. In Table 4.4, we evaluate ODIN on each (ID, OOD) pair using a pre-defined search grid for T and ϵ, and record the best performance (in terms of AUROC). We use the same architecture (ResNet-34) as Lee et al. (2018), and so we also use their search grid, as recorded in Table 4.

However, we also consider the case when $\mathcal{D}_{out}$ is completely unseen. In that case, we used the fixed values for (T, ϵ) as recorded in Table 4.

It is mentioned in the original works by Lee et al. (2018) that higher values of T are to be preferred, but too high

	ID Dataset					
	CIFAR-10		CIFAR-100		SVHN	
OOD Dataset	Full	Low	Full	Low	Full	Low
CIFAR-10	—	—	(1000, 0)	(10, 0)	(100, 0.0005)	(10, 0.005)
CIFAR-100	(10, 0)	(10, 0.0005)	—	—	(10, 0.0005)	(10, 0.005)
SVHN	(10, 0)	(100, 0.05)	(10, 0.01)	(10, 0.1)	—	—
STL-10	—	—	(1000, 0)	(1000, 0)	(10, 0.0005)	(10, 0.005)
CelebA	(1, 0.005)	(1, 0.005)	—	—	(10, 0.002)	(10, 0.01)
TinyImageNet	(1, 0.001)	(10, 0.01)	(10, 0.005)	(100, 0.01)	(10, 0.0005)	(10, 0.005)
LSUN	(1, 0.001)	(10, 0.01)	(10, 0.005)	(10, 0.01)	(1, 0.0014)	(10, 0.005)

Table 5. Optimal hyper-parameter values, $(T_{opt}, \epsilon_{opt})$, for ODIN method. Here for each (ID, OOD) pair, we evaluate ODIN with all pairs of (T, ϵ) from Table 4, and record the pair for which ODIN obtains the highest OOD detection AUROC on that particular (ID, OOD) pair. Notice that optimal hyper-parameters change depending on whether 100% (Full data setting) or 10% (Low data setting) ID training data is available. This table records the optimal values for the final run of our experiments, and we note that a different run might result into different optimal hyper-parameters.

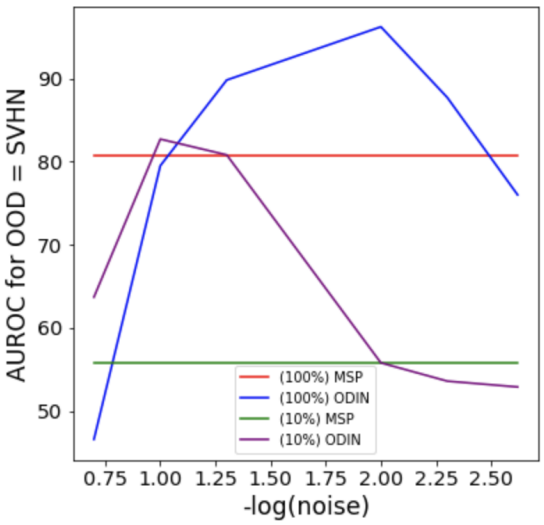

Figure 3. ODIN's sensitivity to noise magnitude, when ID = CIFAR-100, OOD = SVHN, and temperature $T = 1000$. Here we plot the OOD detection AUROC vs $-\log_{10}(\epsilon)$. The horizontal red and green lines are MSP baselines for when 100% and 10% ID train data is available, respectively. Notice that the best performing noise for the full data regime results in very poor performance in the low data regime and vice versa.

values do not yield any additional benefits. Following their work and Hsu et al. (2020), we set $T = 1000$ when we do not have access to OOD samples.

However, unlike temperature, there is no principled way for setting ϵ without access to OOD samples. We see that the performance of ODIN, specially on the low data regime, is highly sensitive to the choice of ϵ, i.e., slight changes in ϵ can harm the performance, and the optimal ϵ depends on too many factors. Even when we keep $\mathcal{D}_{in}$ fixed, the optimal choice of ϵ varies for different choices of $\mathcal{D}_{out}$ and the individual ID training samples available. We demonstrate this in Figure 3, where for (ID, OOD) = (CIFAR-100, SVHN) and $T = 1000$, we see that using the best performing noise for the full data regime results in poor performance in the low data regime and vice versa.

So the optimal ϵ for input pre-processing needs to be tuned for each neural network architecture (Lee et al., 2018), individual (ID, OOD) pair and the examples sampled. Setting ϵ to some arbitrary value above 0 without tuning it properly can make ODIN under-perform MSP. This is why in the setting where we do not have access to OOD samples, we simply set $\epsilon = 0$.

B.4. OOD sample dependent hyper-parameters for Mahalanobis

The hyper-parameter ϵ for Mahalanobis, similar to ODIN, requires OOD samples to be tuned. We use the same architecture (ResNet-34) as Lee et al. (2018), and so we also use their search grid, as recorded in Table 4.

Similar to ODIN, there is no principled way of choosing ϵ, and we set $\epsilon = 0$ in the setting without any OOD samples.

B.5. Optimal hyper-parameters for ODIN

For each (ID, OOD) pair, we evaluate ODIN with all pairs of (T, ϵ) from Table 4, and record the pair for which ODIN obtains the highest OOD detection AUROC on that particular (ID, OOD) pair, $(T_{opt}, \epsilon_{opt})$, in Table 5.

Note that these are the optimal hyper-parameters of our final run based on which we produce the results in this paper, and there can be variations in these numbers in different runs.

	ID Dataset					
	CIFAR-10		CIFAR-100		SVHN	
OOD Dataset	Full	Low	Full	Low	Full	Low
CIFAR-10	—	—	0	0.01	0.005	0.01
CIFAR-100	0	0.01	—	—	0.005	0.005
SVHN	0.01	0.01	0.01	0.05	—	—
STL-10	—	—	0	0.01	0.01	0.01
CelebA	0	0.01	—	—	0.01	0.01
TinyImageNet	0.0024	0.005	0.005	0.01	0.01	0.01
LSUN	0.0024	0.005	0.0024	0.01	0.01	0.01

Table 6. Optimal hyper-parameter values, ϵ_{opt}, for Mahalanobis method. Here for each (ID, OOD) pair, we evaluate Mahalanobis with all values of ϵ from Table 4, and record the value for which Mahalanobis obtains the highest OOD detection AUROC on that particular (ID, OOD) pair.

B.6. Optimal hyper-parameters for Mahalanobis

Similar to Table 5, we record the optimal hyper-paramters for Mahalanobis, ϵ_{opt}, for each (ID, OOD) pair in Table 6.

C. More on the toy datasets

C.1. Algorithms for generating the toy datasets

Here we discuss the pseudocode of a few functions that we later use to generate our toy datasets. Specifically, first we write the pseudocode to create a cluster of data-points in $\mathbb{R}^2$ around (μ_x, μ_y) in Algorithm 1.

Algorithm 1 produceCluster

Input: Mean (μ_x, μ_y), standard deviations (σ_x, σ_y), number of data-points N
Initialize L to be an empty list
for $i = 1$ **to** N **do**
  Pick $x \sim \mathcal{N}(\mu_x, \sigma_x^2)$ and $y \sim \mathcal{N}(\mu_y, \sigma_y^2)$
  Append (x, y) to the list L
end for
Return L

Next we write pseudocode to create a dataset in $\mathbb{R}^2$, where the data-points are spread around a circle. The pseudocode is presented in Algorithm 2.

Algorithm 2 produceRing

Input: Center (C_x, C_y), radius R, number of points N
Initialize L to be an empty list
for $k = 1$ **to** N **do**
  Let $\theta = \frac{2\pi k}{N}$
  Let $x = C_x + R\cos\theta$, $y = C_y + R\sin\theta$
  Append (x, y) to the list L
end for
Return L

C.2. Generating toy dataset 1

We use Algorithm 3 to produce toy dataset 1 in Figure 1. Note that we randomly choose 80% of the ID examples to form dataset $\mathcal{D}_{in}^{train}$, and the other 20% to form dataset $\mathcal{D}_{in}^{test}$.

Algorithm 3 generateToyDataset1

Let ID_Class_1 = produceCluster($\mu_x = -10$, $\mu_y = -10$, $\sigma_x = 0.5$, $\sigma_y = 0.5$, N = 100)
Let ID_Class_2 = produceCluster($\mu_x = -2$, $\mu_y = -2$, $\sigma_x = 0.5$, $\sigma_y = 0.5$, N = 100)
Let OOD_Class = produceCluster($\mu_x = 6.5$, $\mu_y = 6.5$, $\sigma_x = 0.5$, $\sigma_y = 0.5$, N = 40)
Return ID_Class_1, ID_Class_2, OOD_Class

C.3. Generating toy dataset 2

We use Algorithm 4 to produce toy dataset 2 in Figure 2. We randomly choose 50% of the ID examples to form dataset $\mathcal{D}_{in}^{train}$, and the other 50% to form dataset $\mathcal{D}_{in}^{test}$.

Algorithm 4 generateToyDataset2

Let ring_1 = produceRing($C_x = -5$, $C_y = -5$, $R = 2.5$, $N = 9$
Let cluster_1 = produceCluster($\mu_x = -0.8$, $\mu_y = -0.8$, $\sigma_x = 0.2$, $\sigma_y = 0.2$, N = 6)
Let ID_Class_1 = ring_1 $\cup$ cluster_1
Let ID_Class_2 = produceRing($C_x = 8$, $C_y = 8$, $R = 2.5$, $N = 15$)
Let OOD_Class = produceCluster($\mu_x = 0$, $\mu_y = 0$, $\sigma_x = 0.3$, $\sigma_y = 0.3$, N = 10)
Return ID_Class_1, ID_Class_2, OOD_Class

D. More on Pairwise OOD Detection (POD)

D.1. Scoring mechanism

Here we consider a more general setting for the Pairwise OOD Detection (POD) method than that described in Section 3, and describe why we choose the special setting in Section 3 even though there were other options available.

Consider the problem description in Section 2. Let $h(x) \in \mathbb{R}^d$ be the neural network's penultimate layer's representation for test sample x. Suppose each of the C ID classes has M training examples i.e., $\{\{p_{ij}\}_{j=1}^{M}\}_{i=1}^{C}$ where p_{ij} is the j-th example from the i-th ID class (Here we consider all ID train examples, however in practice some smaller value of M is chosen to reduce computation costs). Let $g_M : \mathbb{R}^M \to \mathbb{R}$ and $g_C : \mathbb{R}^C \to \mathbb{R}$ be two accumulator functions. For any $x \in \mathcal{X}$, we define the following:

$$dist(x, p_{ij}) = ||h(x) - h(p_{ij})||_2^2 \tag{13}$$

$$S^i_{POD}(x) = g_M(\{dist(x, p_{ij})\}^M_{j=1}) \quad (14)$$

Here instead of taking the average of $dist(x, p_{ij})$ over $j = 1$ to M, we use the general function g_M. Similarly, we define,

$$S_{POD}(x) = g_C(\{S^i_{POD}(x)\}^C_{i=1}) \quad (15)$$

We note that there are a couple intuitive choices for (g_C, g_M) such as (min, average), (average, average) and (min, min). In all our experiments, we use (min, average). Here we provide a toy dataset first showing why we make this choice, next we provide empirical evidence supporting our choice.

D.2. Toy dataset differentiating between different POD scoring schemes

Consider toy dataset 2 in Figure 2. In the original example, we consider (g_C, g_M) = (min, min) and show that distance based methods perform poorly on the dataset. Here however, we consider different intuitive choices for (g_C, g_M) and record their performances in Table 7.

Choice for (g_C, g_M)	OOD detection AUROC
(min, min)	32.5%
(min, average)	100.0%
(average, average)	0%

Table 7. Performance of POD method (OOD Detection AUROC) on toy dataset 2 under different choices for (g_C, g_M).

We have already discussed that the ID outliers force (min, min) to perform poorly. However, (min, average) does not have that flaw, under the assumption that outliers constitute only a small portion of the ID dataset. This is because averaging over the training examples in individual classes negates the effect of the few ID outliers. (average, average) does the worst here because the OOD class is actually between the two ID classes. For a test example belonging to ID class 1, the distances to ID class 2 examples will be much larger and vice versa, making the average distance large for ID test examples — however, the OOD samples are between both classes, and so the average distance is smaller for OOD samples.

D.3. Empirical evidence for the choice of POD scoring scheme

We also observe empirically in our experiments that (min, average) performs most consistently across different (ID, OOD) pairs and full vs low data experiments. We record the OOD detection performance of POD method for different choices of (g_C, g_M) in Table 8 when ID = CIFAR-10 and 10% training data is used.

	OOD detection AUROC for choice of (g_C, g_M)		
OOD Dataset	(min, average)	(average, average)	(min, min)
CIFAR-100	**77.9**	77.4	77.1
SVHN	**76.2**	44.1	71.0
CelebA	**77.7**	75.9	74.4
TinyImageNet	73.0	**75.9**	76.2
LSUN	77.4	**82.7**	80.7

Table 8. OOD detection performance of POD method with ID = CIFAR-10 and low data setting (10% ID train data used) for different choices of (g_C, g_M). Bold numbers represent superior result on a particular (ID, OOD) pair.

However, we note that different (g_C, g_M) choices work better on different (ID, OOD) pairs, and the inconsistency among these results are also significant.. This can be considered a hyper-parameter for POD method, and we suggest investigating more into how conceptually/empirically the choice of (g_C, g_M) affects OOD detection. In the more structured setting where one has access to small number of OOD examples, one may also follow ODIN (Liang et al., 2018) or Mahalanobis (Lee et al., 2018) and use OOD samples to determine the choice for (g_C, g_M).

E. Fine-tuning on pairwise classification task

Inspired by the OOD detection performance of the POD method, we also experiment with fine-tuning a regular classifier on the pairwise classification task. We call this method POD + Fine-tune. We will first discuss the algorithm for fine-tuning, and then discuss the findings.

E.1. Algorithm for fine-tuning

Let us assume that the penultimate layer of the regular neural network classifier, h, produces an embedding in $\mathbb{R}^d$, i.e., for any $x \in \mathcal{X}$, we have $h(x) \in \mathbb{R}^d$. For fine-tuning, we introduce a learnable parameter $w \in \mathbb{R}^d$. Given an image pair $(x_1, x_2) \in \mathcal{X} \times \mathcal{X}$, and label $y_p = 0$ if x_1 and x_2 belong to the same class, or $y_p = 1$, we define the model output as follows:

$$Pairwise(x, y) = \sum_{j=1}^{d} w_j (h(x_1)_j - h(x_2)_j)^2 \quad (16)$$

We fine-tune f and w with binary cross entropy loss between y_p and $Pairwise(x, y)$ using Algorithm 5.

Algorithm 5 Fine Tune POD

Input: Regular classifier f, Train dataset $\mathcal{D}_{train}$, number of epochs N_{epoch}, number of data-pairs per epoch N_{pairs}
for $i = 1$ **to** N_{epoch} **do**
  Initialize $\mathcal{D}_p$ to be an empty dataset
  Initialize w to be a vector in $\mathbb{R}^d$
  for $j = 1$ **to** N_{pairs} **do**
    if j is even **then**
      Randomly choose a class c in $\mathcal{D}_{train}$
      Randomly choose data-points $(x_1, y_1), (x_2, y_2) \in \mathcal{D}_{train}$ such that $x_1 \neq x_2$ and $y_1 = y_2 = c$
      Append $((x_1, x_2), 0)$ to $\mathcal{D}_p$
    else
      Randomly choose classes $c_1 \neq c_2$ in $\mathcal{D}_{train}$
      Randomly choose data-point $(x_1, y_1) \in \mathcal{D}_{train}$ such that $y_1 = c_1$
      Randomly choose data-point $(x_2, y_2) \in \mathcal{D}_{train}$ such that $y_2 = c_2$
      Append $((x_1, x_2), 1)$ to $\mathcal{D}_p$
    end if
  end for
  Fine-tune f and w on dataset $\mathcal{D}_p$
end for

Following the setting described for POD method in Section 3, we define:

$$S^i_{POD+FT}(x) = \frac{1}{M}\sum_{j=1}^{M}\sum_{k=1}^{d} w_k (h(x)_k - h(p_{ij})_k)^2 \quad (17)$$

Finally, we saw in Appendix D that (min, average) is the best scoring scheme for POD. Surprisingly, (average, average) is the overall best scoring scheme for POD + Fine-tune (See empirical results for ID = CIFAR-10 and the full data setting on Table 9). We suggest further investigations into why this occurs. With this in mind, we define the score function for POD + Fine-tune as follows:

$$S_{POD+FT}(x) = \frac{1}{C}\sum_{i=1}^{C} S^i_{POD,FT}(x) \quad (18)$$

And use this as a heuristic for OOD detection as described in Section 2.

E.2. Observation

We notice that POD + Fine-tune actually does worse than POD in most of the cases (See Table 11 and 12). Even though this is surprising, we note that it is similar to the findings of Reed et al. (2021) and Kolesnikov et al. (2019). Kolesnikov et al. (2019) finds that directly training a full network on rotation prediction harms the correlation between rotation prediction and supervised performance. Additionally, Reed et al. (2021) shows that training a 2-layer network on rotation prediction improves the performance on the rotation prediction task but hurts the correlation between rotation accuracy and supervised evaluation of learned representations. Similarly, in our work we see that fine-tuning a whole ResNet-34 (He et al., 2016) network on pairwise prediction task improves pairwise accuracy — but since the network can learn its own representations, this ends up hurting OOD detection performance in most cases. This may require more study in the future.

	OOD detection AUROC for choice of (f_C, f_M)		
OOD Dataset	(min, average)	(average, average)	(min, min)
CIFAR-100	88.3	87.1	**88.6**
SVHN	93.8	**97.0**	92.3
CelebA	73.0	**77.6**	72.1
TinyImageNet	91.2	**92.6**	92.3
LSUN	93.3	**94.5**	94.0

Table 9. OOD detection performance of POD + Fine-tune method with ID = CIFAR-10 and full data setting (100% ID train data used) for different choices of (g_C, g_M) (See Appendix D.1 for details of scoring mechanism). Bold numbers represent superior result on a particular (ID, OOD) pair.

F. ID classification accuracy

We also record the in-distribution classification accuracy of MSP, POD and POD + Fine-tune methods. Given $x \in \mathcal{X}$, we define —

$$Pred_{MSP}(x) = \underset{i\in\{1,\cdots,C\}}{\text{argmax}}\ S^i_{soft}(x; T = 1) \quad (19)$$

$$Pred_{POD}(x) = \underset{i\in\{1,\cdots,C\}}{\text{argmin}}\ S^i_{POD}(x) \quad (20)$$

$$Pred_{POD+FT}(x) = \underset{i\in\{1,\cdots,C\}}{\text{argmin}}\ S^i_{POD+FT}(x) \quad (21)$$

where $S^i_{soft}(x; T)$, $S^i_{POD}(x)$ and $S^i_{POD}(x)$ are defined in Equations 1, 5 and 17, respectively. The results are summarized in Table 10.

	MSP		POD		POD + FT	
ID Dataset	Full	Low	Full	Low	Full	Low
CIFAR-10	95.75	82.21	95.77	81.74	94.42	83.03
CIFAR-100	79.26	37.37	79.17	36.63	72.56	37.55
SVHN	96.12	91.95	96.09	91.93	95.98	91.26

Table 10. Classification accuracy on $\mathcal{D}^{test}_{in}$, for MSP, POD and POD + FT methods.

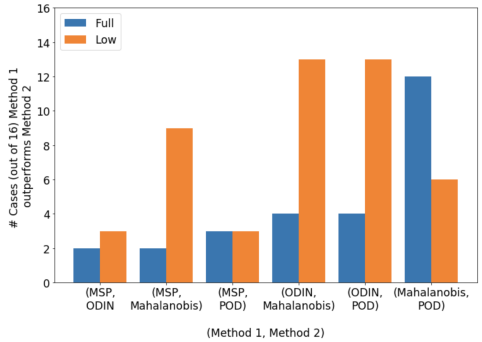

Figure 4. Bar plot showing the number of ID-OOD dataset pairs (out of 16) on which Method 1 outperforms Method 2 on OOD detection by AUROC, without any access to OOD samples for training/hyper-parameter tuning. Notice the change in performance due to the change in ID training dataset size. In 4 out of 6 cases, one method outperforming another on the majority of (ID, OOD) pairs in the full data setting does not do so in the low data regime.

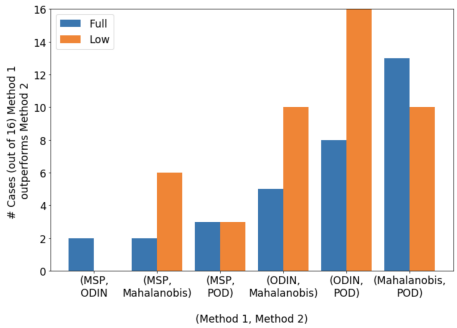

Figure 5. Bar plot showing the number of ID-OOD dataset pairs (out of 16) on which Method 1 outperforms Method 2 on OOD detection by AUROC, where ODIN and Mahalanobis have access to OOD samples for training/hyper-parameter tuning. Despite inconsistency being lower compared to Figure 4. However, there is still inconsistency, for example the performance order on the majority of 16 cases between ODIN and Mahalanobis switches from low to full data setting.

We see that POD maintains very similar classification performance as MSP in all cases, however, POD + FT has performance deterioration in some cases. This might be because the neural net prioritizes learning embeddings that improves pairwise classification performance at the cost of regular classification performance during fine-tuning.

G. Empirical results

For full data setting (100% ID train data is available), we record our detailed results in Table 11. Similarly we record our low data setting (10% ID train data is available) experiment results in Table 12. In both cases, we record the (AUROC, FNR@95) pair for each (ID, OOD) pair and each method. We compare between two methods based on their OOD detection AUROC first, and then break any ties using FNR@95.

H. Full vs low data regime

One major inconsistency that we notice is between the full (100%) and low (10%) data regimes. This inconsistency is most pronounced when there is no access to OOD samples (see 4), and we see that in 4 out of 6 total (method 1, method 2) pairs, if method 1 outperforms method 2 on the majority of (ID, OOD) pairs in the full data setting, than the situation reverses on low data setting. This is concerning, because in a real world setting, it would be hard to know if we have enough data to capture the diversity in the ID dataset. Access to large amount of ID data is often difficult to have — in scenarios like monitoring bio-diversity (Høye et al., 2021) or classifying diseases (Prabhu et al., 2018) — due to a variety of reason, most notably, difficulty in data collection and regulations due to confidentiality.

We see that without large number of ID data, a method performing well may not do so. This problem is mitigated a little bit if there is access some OOD examples to tune hyper-parameters, as we see in Figure 5. However, there is still inconsistency. For example, with access to OOD samples, ODIN does better than Mahalanobis on only 5 out of 16 (ID, OOD) pairs in the full data setting; however, in the low data setting, ODIN does better than Mahalanobis on 10 out of 16 pairs. This shows that we should consider another factor while discussing the OOD detection problem, namely the size of the ID dataset.

I. Relation between ID classification and OOD detection performance

In an earlier iteration of the work, we experimented to see if a neural network's in-distribution classification accuracy and OOD detection performance are correlated. This would imply that deploying a neural network with higher in-distribution accuracy will also make it effective at OOD detection.

However, this turned out to be not true in general. Here we report one such experiment. We train a WideResNet (Zagoruyko & Komodakis, 2016) architecture with depth 40 and widen factor 2 on the CIFAR-10 dataset for varying number of epochs, similar to Hendrycks et al. (2019). In particular, we train neural networks for 50, 75, 100, 200, 300, 400 and 500 epochs, with batch size 128, SGD with Nesterov momentum, initial learning rate 0.1 and

		OOD Detection Performance — (AUROC, FNR@95)						
ID	OOD	MSP	ODIN (W/O OOD)	ODIN (With OOD)	Maha (W/O OOD)	Maha (With OOD)	POD	POD + FT
CIFAR-10	CIFAR-100	(86.7, 68.9)	(85.6, 77.7)	(85.8, 76.3)	**(89.7, 39.8)**	**(89.7, 39.8)**	(89.3, 40.9)	(87.1, 62.1)
	SVHN	(95.0, 13.7)	(95.7, 12.9)	(95.7, 12.9)	(95.1, 11.9)	**(98.2, 9.4)**	(94.7, 15.0)	(97.0, 13.6)
	CelebA	(74.3, 78.4)	(67.6, 87.0)	(72.5, 79.6)	(76.0, 69.6)	(76.0, 69.6)	(77.0, 63.1)	**(77.6, 58.3)**
	TinyImageNet	(90.0, 53.8)	(90.4, 63.1)	(90.8, 62.3)	(93.1, 23.0)	**(95.3, 21.0)**	(92.3, 29.3)	(92.6, 38.0)
	LSUN	(93.2, 21.3)	(94.1, 23.7)	(94.5, 26.3)	(94.9, 15.6)	**(97.1, 11.5)**	(94.1, 20.3)	(94.5, 25.4)
CIFAR-100	CIFAR-10	(78.1, 64.5)	**(78.5, 66.6)**	**(78.5, 66.6)**	(75.3, 71.2)	(75.3, 71.2)	(76.0, 76.9)	(76.2, 69.6)
	SVHN	(80.7, 50.5)	(81.2, 51.1)	**(96.2, 17.8)**	(88.1, 41.4)	(94.3, 23.8)	(83.7, 48.3)	(76.7, 57.6)
	STL-10	(79.4, 59.4)	**(79.8, 60.8)**	**(79.8, 60.8)**	(77.6, 64.1)	(77.6, 64.1)	(78.1, 68.0)	(77.4, 65.9)
	TinyImageNet	(81.2, 53.3)	(83.1, 51.2)	**(90.1, 41.6)**	(83.7, 50.0)	(89.3, 41.9)	(84.3, 48.4)	(87.3, 39.2)
	LSUN	(82.7, 48.6)	(84.7, 46.8)	(91.5, 35.1)	(86.9, 41.8)	**(91.8, 32.4)**	(86.8, 42.1)	(90.1, 33.7)
SVHN	CIFAR-10	(95.5, 12.2)	(95.9, 12.0)	(96.0, 11.8)	(96.3, 11.7)	**(97.8, 8.1)**	(96.2, 11.9)	(93.8, 34.5)
	CIFAR-100	(95.0, 13.3)	(95.5, 13.3)	(95.5, 13.3)	(96.0, 12.4)	**(97.4, 9.1)**	(95.9, 12.7)	(93.0, 39.9)
	STL-10	(95.7, 11.4)	(96.1, 11.1)	(96.2, 10.9)	(96.4, 11.1)	**(98.0, 8.4)**	(96.3, 11.3)	(94.7, 26.5)
	CelebA	(96.5, 10.0)	(97.0, 9.6)	(97.5, 9.8)	(97.0, 9.9)	**(98.7, 5.3)**	(96.8, 10.6)	(94.1, 29.5)
	TinyImageNet	(95.2, 12.5)	(95.6, 12.5)	(95.7, 12.2)	(96.1, 11.5)	**(98.1, 7.8)**	(96.1, 11.8)	(93.7, 34.1)
	LSUN	(94.8, 13.5)	(95.2, 13.7)	(95.3, 13.8)	(96.0, 12.3)	**(98.1, 7.7)**	(95.9, 12.6)	(93.1, 37.4)

Table 11. Detailed results for full data setting (100% ID training data available). For each (ID, OOD) pair and for each method, we report the (AUROC, FNR@95) pair. Bold results represent the superior result for a particular (ID, OOD) pair. We compare between two methods based on their OOD detection AUROC first, and then break any ties using FNR@95. All numbers are average of 3 runs.

		OOD Detection Performance — (AUROC, FNR@95)						
ID	OOD	MSP	ODIN (W/O OOD)	ODIN (With OOD)	Maha (W/O OOD)	Maha (With OOD)	POD	POD + FT
CIFAR-10	CIFAR-100	(77.8, 59.3)	(79.0, 61.6)	(79.1, 61.7)	(72.7, 70.7)	(73.5, 72.6)	(77.9, 63.4)	**(80.2, 59.2)**
	SVHN	(60.5, 84.0)	(55.5, 86.2)	**(93.5, 31.3)**	(89.7, 39.1)	(93.4, 31.6)	(76.2, 51.7)	(76.2, 59.1)
	CelebA	(75.4, 57.2)	(75.0, 60.4)	**(78.4, 57.5)**	(65.9, 74.7)	(68.9, 68.2)	(77.7, 55.6)	(74.6, 50.1)
	TinyImageNet	(74.7, 65.2)	(77.6, 63.4)	**(88.7, 43.5)**	(80.9, 57.3)	(87.9, 48.0)	(73.0, 73.0)	(74.6, 71.7)
	LSUN	(81.2, 47.7)	(84.8, 44.4)	**(93.9, 24.2)**	(82.0, 49.9)	(90.0, 36.6)	(77.4, 59.5)	(81.7, 55.1)
CIFAR-100	CIFAR-10	(61.0, 82.1)	(62.0, 82.2)	**(62.1, 82.2)**	(55.3, 87.1)	(56.0, 87.1)	(61.4, 80.8)	(61.0, 82.9)
	SVHN	(55.8, 82.8)	(52.8, 83.4)	**(82.7, 54.4)**	(82.3, 66.0)	(82.7, 56.1)	(62.0, 82.7)	(49.8, 86.6)
	STL-10	(62.0, 82.1)	**(62.5, 81.2)**	**(62.5, 81.2)**	(54.9, 87.0)	(55.2, 86.8)	(61.4, 81.2)	(61.2, 82.3)
	TinyImageNet	(65.0, 80.7)	(66.8, 79.2)	**(69.6, 78.1)**	(52.0, 87.0)	(59.4, 83.2)	(65.1, 78.4)	(62.7, 80.2)
	LSUN	(66.1, 79.3)	(68.4, 76.6)	**(70.6, 75.1)**	(50.9, 86.5)	(58.0, 83.6)	(65.5, 77.6)	(60.6, 83.1)
SVHN	CIFAR-10	(92.4, 22.0)	(93.2, 21.6)	(94.9, 18.4)	(92.4, 21.5)	**(95.3, 16.8)**	(92.4, 21.9)	(91.6, 25.9)
	CIFAR-100	(92.1, 22.7)	(92.8, 22.8)	(94.2, 20.4)	(92.1, 22.5)	**(94.6, 18.8)**	(92.1, 22.4)	(91.6, 26.5)
	STL-10	(92.5, 21.6)	(93.5, 21.1)	(95.2, 17.6)	(92.4, 21.8)	**(95.6, 16.3)**	(92.6, 21.4)	(91.9, 25.4)
	CelebA	(94.0, 17.3)	(95.0, 16.4)	(97.4, 10.0)	(93.8, 17.9)	**(97.4, 9.8)**	(94.2, 17.8)	(91.9, 26.0)
	TinyImageNet	(92.5, 21.4)	(93.3, 21.1)	(95.3, 17.5)	(92.5, 21.0)	**(95.7, 16.1)**	(92.5, 21.2)	(92.1, 25.0)
	LSUN	(92.2, 22.1)	(92.9, 21.9)	(95.3, 17.4)	(92.1, 22.0)	(95.8, 15.8)	(92.5, 21.4)	(91.8, 25.4)

Table 12. Detailed results for low data setting (10% ID training data available). For each (ID, OOD) pair and for each method, we report the (AUROC, FNR@95) pair. Bold results represent the superior result for a particular (ID, OOD) pair. We compare between two methods based on their OOD detection AUROC first, and then break any ties using FNR@95. All numbers are average of 3 runs.

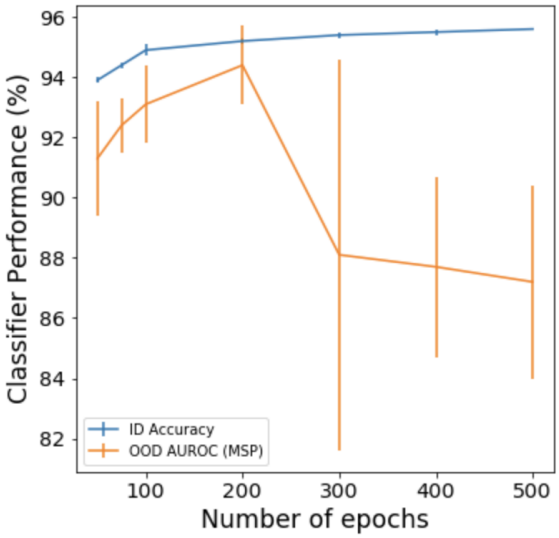

Figure 6. ID classification accuracy and OOD detection AUROC vs number of epochs, for (ID, OOD) = (CIFAR-10, SVHN). For each $N_{epoch} \in \{50, 75, 100, 200, 300, 400, 500\}$, we train three WideResNet-40-2 (Zagoruyko & Komodakis, 2016) networks for N_{epoch} epochs, and plot the average values. The standard deviations are used for error bars.

cosine annealing learning rate (Loshchilov & Hutter, 2017). We notice that the neural network always improves on ID classification accuracy as the number of epochs increase. However, the OOD detection performance for (ID, OOD) = (CIFAR-10, SVHN) seems to drop once the number of epochs increase beyond 200. For a particular number of epochs, we train 3 neural networks and plot the average values in Figure 6, with the standard deviations as error bars.

Additionally, Figure 6 shows that increase in ID classification accuracy does not always translate to an increase in OOD detection performance. Furthermore, we see that the standard deviation on OOD detection AUROC is much larger than the standard deviation for ID classification accuracy — so model performance on OOD detection task fluctuates more than its performance on ID classification task.