Intra-Dataset Outlier Exposure: Mitigating the Misclassification of Open Set Objects in a Colorectal Cancer Detecting Neural Network

Olina Mukherjee
Taylor Allderdice High School
2409 Shady Ave, Pittsburgh, PA

Advised by:
Dr. Ben Erichson
Robust Deep Learning Group at the International Computer Science Institute (ICSI)
2150 Shattuck Ave. #250, Berkeley, CA 94704

Abstract

Neural networks typically operate with a closed-world assumption. This implies that they are only designed to classify objects that they were trained on, or in-distribution (ID) objects. However, when deployed in the real world, networks inevitably come across objects that they were *not* trained on, or out-of-distribution (OOD) objects. In these situations, the closed-world assumption leaves networks ill-prepared, resulting in the misclassification of OOD objects. In safety-critical applications, like cancer diagnosis, such misclassifications can have fatal consequences. This study aims to train a neural network which mitigates the misclassification of open-set objects, a subset of OOD objects which belong to the in-distribution dataset. When evaluated on the MNIST and CIFAR-10 datasets, the proposed *Intra-Dataset Outlier Exposure* (*IDOE*) method outperforms almost all state-of-the-art methods. Additionally, when tested on the PathMNIST dataset of healthy and cancerous colorectal tissues, *IDOE* achieved an AUROC score of 94.8%, demonstrating its high efficacy in medical diagnostic applications.

1. Introduction

When machine learning techniques were applied to neural networks initially, they were primarily tested on the objects they were trained to detect. These methods have yielded impressive results, so, many of these recent developments are being deployed in real-world applications, including autonomous vehicles and medical diagnostic algorithms. Neural networks have seen great success in many of these applications [1]. Nevertheless, they exhibit sensitivities to various factors, including adversarial perturbations, corruptions like blurry or noisy images, and out-of-distribution (OOD) objects [2].

OOD objects pose a particularly challenging threat to neural networks. Most networks accurately classify in-distribution (ID) objects, which they are trained on. However, networks tend to misclassify OOD objects, which they are not trained on [2]. These misclassifications can be fatal in safety-critical applications like medical diagnosis and autonomous vehicle navigation, especially since these applications often involve outliers like emerging ailments and newly installed speed bumps.

The field of generalized OOD detection [3] includes OOD detection [2] as well as open-set recognition (OSR) [4]. In the subfield of OSR, trained ID objects are referred to as closed-set objects and outliers are referred to as open-set objects. Functionally, both OOD detection and OSR are concerned with creating robust networks that accurately classify trained objects (either ID or closed-set objects) while also recognizing outliers (either OOD or open-set objects) [3]. However, OSR requires networks to recognize open-set outlier objects which lie within the *same* distribution as trained closed-set objects, while OOD detection requires networks to identify OOD outliers which typically belong to a distribution that is distinctly *different* from the in-distribution. Consider a scenario where a network trained to classify images of slides containing cancerous tissue samples encounters an image of a slide with tissue debris, dead or damaged tissue, on it. Since the outlier, tissue debris, exists in the same slide environment as the trained cancerous tissues, it lies within the same distribution. Therefore, this situation requires OSR where the tissue debris is an open-set object, and the cancerous tissues are closed-set objects. A scenario where the same cancer-detecting network encounters a handwritten digit requires OOD detection, since digits and tissues exist in completely different environments. In this case, the digit is an OOD object, and the tissues are ID objects. In general, OSR tasks tend to be more challenging than OOD tasks because the differences between open and closed-set objects are often more subtle than the differences between OOD and ID objects, as described in the examples above.

To prevent the misclassification of open-set objects, this study proposes *Intra-Dataset Outlier Exposure (IDOE)*, a method where neural networks are exposed to example open-set objects while they are trained. These trained examples allow networks to generalize, classifying all outliers as open-set. The contributions of this paper are:

1. Applying outlier exposure to open set problems with IDOE. By training networks on outlier examples, networks learn the intrinsic differences between open and closed set objects.
2. Deploying IDOE on the PathMNIST [5] dataset of colorectal tissues. This allowed for the evaluation of IDOE in a real-world application where open-set outliers are very visually similar to closed-set trained objects; more so than current OSR datasets account for.

The rest of this paper is organized as follows. In section 2, works related to the fields of OOD detection and OSR are reviewed. In section 3, the motivation for the proposed IDOE method is described. In section 4, the details of IDOE's implementation are elaborated on. In section 5, experimental results and analyses are provided. Finally, in section 6, conclusions are discussed.

2. Related Works

Most open-set recognition (OSR) approaches compare encountered objects to the closed-set objects they were trained on. OpenMax [6] fits trained closed-set objects to a Weibull distribution and classifies encountered objects that lie significantly outside this distribution as open-set objects. K-Nearest Neighbors (KNN) [7] classifies objects as open-set if they have structural features that are significantly different from the closed-set objects' features. When applied to the MNIST dataset of handwritten digits [8], OpenMax and KNN achieve Area Under the Receiver Operating Characteristic curve (AUROC) [9] scores of 97.3% and 97.5%. When applied to the CIFAR-10 dataset of animals and vehicles [10] OpenMax and KNN achieve AUROC scores of 84.2% an 86.9% respectively.

On the other hand, Outlier Exposure, proposed by Hendrycks *et al*. [11] trains networks with an auxiliary class of out-of-distribution (OOD) object examples. This class is an amalgamation of a whole OOD dataset (a dataset that is distinctly different from the ID dataset). Outlier Exposure is referred to as *Classical Outlier Exposure (COE)* in this study to contrast it with the proposed *Intra-Dataset Outlier Exposure (IDOE)* method. When trained with the 300K Random Images dataset [11] as its auxiliary class, COE achieves Area Under the Precision Recall curve (AUPR) [12] scores of 97% and 97.8% on MNIST [8] and CIFAR-10 [10] respectively. This training method is depicted in Figure 1 with MNIST as the ID dataset. By exposing networks to a wide range of outlier characteristics using the outlier examples in the OOD class, COE trains networks to classify objects as OOD if they do not closely resemble any ID objects but share characteristics with the example outliers in the OOD class.

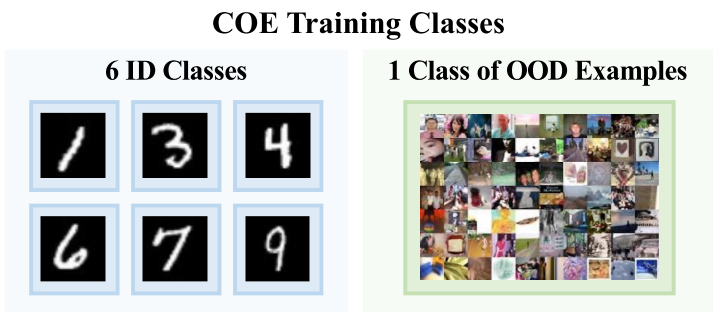


Figure 1. *COE has an auxiliary OOD class in addition to its ID classes. The OOD class is an amalgamation of the 300K Random Images dataset. MNIST images Source: [8] 300K Random Image Source: [11]*

3. Motivation for Developing IDOE

Though Classical Outlier Exposure (COE) has succeeded in OOD tasks, as its high scores show, it has not been applied to OSR tasks. However, since it exposes networks to outlier examples during training, enabling networks to recognize and generalize the intrinsic characteristics of open-set objects, it is likely to have an advantage over state-of-the-art OSR methods which focus on measuring differences between closed and open-set objects. This is particularly pertinent for datasets with subtle differences between the closed and open-sets and therefore demand a more nuanced understanding of these sets. Medical datasets like PathMNIST [5], which consists of visually similar colorectal tissues, fall in this category.

However, if all the example open-set objects belong to 300K Random Images [11] and all the closed-set objects belong to the MNIST dataset of handwritten digits [8], the network may learn that it should only classify objects that are similar to the 300K Random Images as open-set objects. This would cause the network to misclassify open-set digits as closed-set since open-set digits are more similar to closed-set digits than they are to random objects, as depicted in Figure 2.

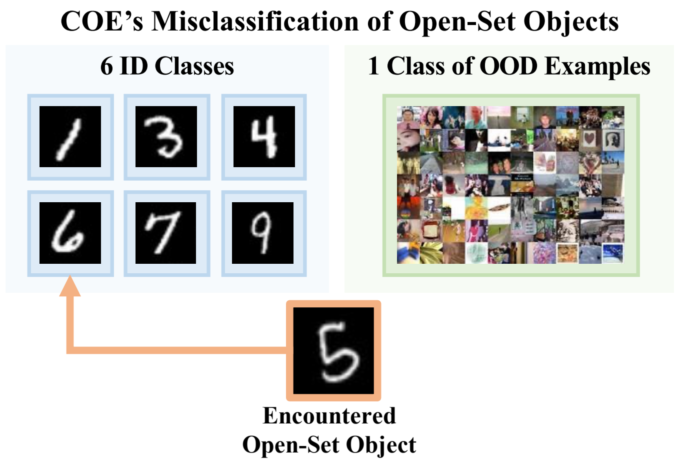


Figure 2. *Open-set objects like "5" are more similar to ID/closed-set digits than they are to random OOD examples. This likely leads COE to misclassify open-set objects like the "5" as ID/closed-set. MNIST images Source: [8] 300K Random Image Source: [11]*

Therefore, this study proposes *Intra-Dataset Outlier Exposure (IDOE)*. IDOE applies outlier exposure to OSR using an auxiliary class of outliers from the closed-set dataset instead of the 300K Random Images dataset. Drawing the auxiliary open-set examples and closed-set objects from the same dataset, IDOE ensures that the primary distinctions between open and closed-set objects are their inherent attributes, rather than superficial characteristics, like background or image quality. This allows the network to learn to recognize the various common characteristics of open-set objects and accurately classify objects that have any of these characteristics and do not closely resemble any closed-set objects, as open-set as shown in Figure 3.

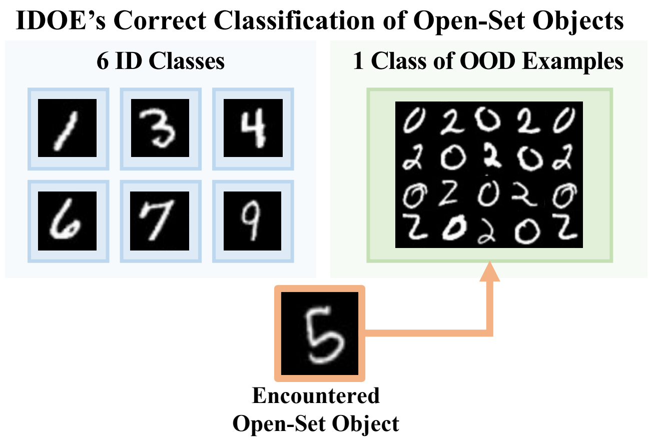


Figure 3. *The main distinction between open and closed-set objects is their inherent attributes. Since the open-set "5" doesn't closely resemble any of the closed-set objects and it does share the black background, white text, and image quality traits with the open set examples, it is likely to be classified as open-set. MNIST images Source: [8]*

Using benchmarks proposed by Jingkang Yang *et al.* in "OpenOOD: Benchmarking Generalized Out-of-Distribution Detection" [13], IDOE is evaluated on several OSR tasks and compared to state-of-the-art methods, including OpenMax [6], KNN [7], and COE [11]. It is additionally deployed on the PathMNIST dataset of colorectal cancerous and healthy tissues [5] to evaluate its efficacy in medical diagnostic applications where open-set objects are quite common and their misclassification can be fatal.

4. Methodology

This section describes how Intra-Dataset Outlier Exposure (IDOE) is applied in a neural network and experimentally evaluated.

4.1 Proposed Framework

Consider a k-class neural network, f, that seeks to learn a nonlinear function $h: \mathcal{X} \rightarrow \mathcal{Y}$ which maps a set of input images, $\mathcal{X}$, to a set of output class labels, $\mathcal{Y} = \{1, \dots, k\}$. With a training set of m examples

$(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ is the correct output for $h(x_i)$, f learns the parameters θ by minimizing empirical risk with loss function ℓ:

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} \ell(h_\theta(x_i), y_i).$$

The novelty of IDOE is the introduction of an auxiliary class of example open-set objects, o. As a result, $\mathcal{Y}' = \{1, \ldots, k, o\}$ is used instead of $\mathcal{Y}$. f then learns that $h(x_{test}) = o$ if $h(x_{test}) \notin \mathcal{Y}$. In other words, IDOE enables networks to classify outliers that were not seen during training as belonging to the open-set class. This way, outliers are no longer misclassified as closed set objects.

4.2 Experimental Setup

In evaluating IDOE, this study follows the benchmarks described in [13]. IDOE was deployed on two standard datasets: MNIST [8], containing 70,000 images of handwritten digits, and CIFAR-10 [10], containing 60,000 images of animals and vehicles. Both datasets have 10 classes. As per the benchmarks, a 6/4 split was used, where six classes were designated as closed-set classes, and the remaining four classes were designated as open-set classes. With IDOE, two of those classes were amalgamated to form the auxiliary class of open-set examples during training, as shown in Figure 4, leaving two open-set classes for IDOE to be evaluated on.

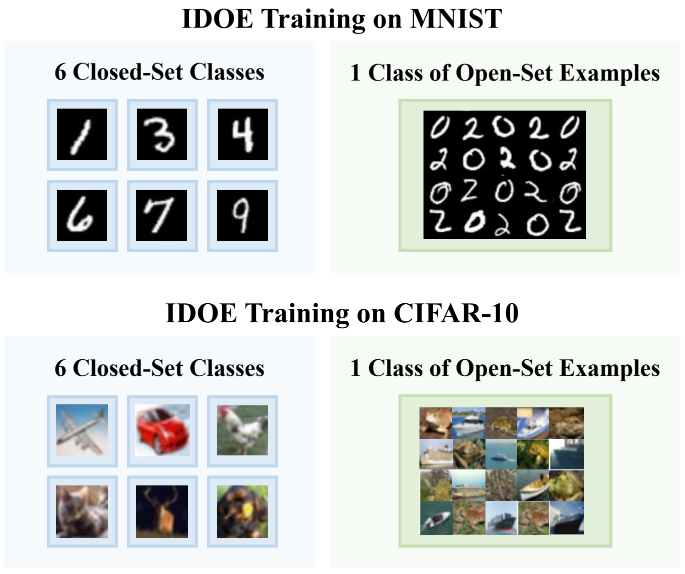

IDOE Training on MNIST
6 Closed-Set Classes
1 Class of Open-Set Examples
IDOE Training on CIFAR-10
6 Closed-Set Classes
1 Class of Open-Set Examples

Figure 4. *IDOE trains networks using six closed and two open-set. MNIST images Source: [8] CIFAR-10 images Source: [10]*

The LeNet-5 [14] and ResNet18 [15] models were used as base network architectures for the MNIST and CIFAR-10 datasets respectively. LeNet-5 is one of the earliest image classification neural network architectures and it was in fact designed for handwritten digit classification. As a result, it has been evaluated on the MNIST dataset extensively. As one of the simplest neural network architectures, it consists of 7 layers: two sets of convolutional and pooling layers followed by flatten and dense layers as shown in Figure 5.

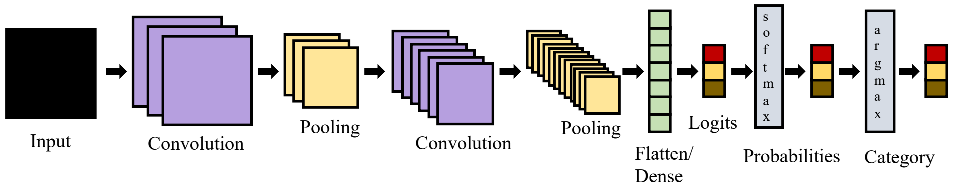


Figure 5. *The LeNet-5 architecture is made up of alternating convolution and pooling layers that recognize features and traits of trained ID/closed-set objects.*

ResNet18 is more complex than LeNet-5, with 18 sets of convolution layers as shown in Figure 6. As a result, this model is better suited for more complex datasets, like CIFAR10.

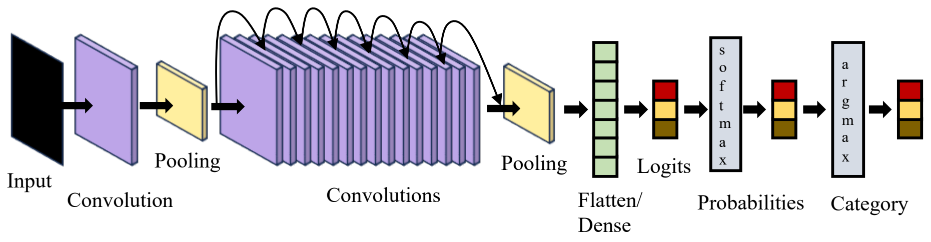


Figure 6. *ResNet18 is deeper than LeNet with 17 convolutional layers and uses skip connections (depicted by the black arrows) to maintain the original image's semantic value despite repeated convolutions. This enables more detailed image analysis for complex datasets.*

To prevent overfitting each network was trained using SGD optimizer, a learning rate of 0.1, momentum of 0.9, and weight decay of 0.0005 for 100 epochs, as per [13]. Additionally, the Area Under the Receiver Operating Characteristic curve (AUROC) [9] evaluation metric was used to evaluate and compare IDOE's performance to state-of-the-art methods. Accounting for true and false positive rates, as shown in Figure 7, this metric holistically describes models' abilities to distinguish between open and closed-sets. Higher AUROC scores are better and random classifiers have an AUROC of 50%.

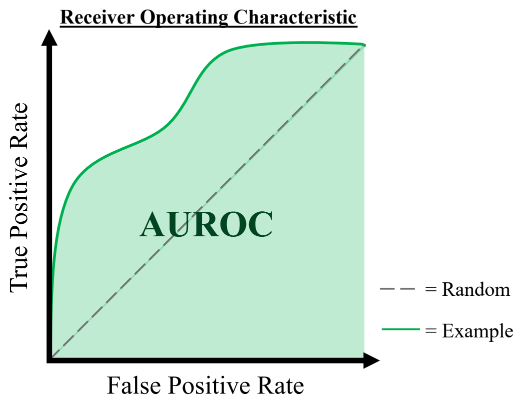


Figure 7. *Area Under the Receiver Operating Characteristic curve evaluation metric.*

Lastly, IDOE was used to train multiple networks with different closed and open-set configurations for each dataset. For example, one digit-classifying network may be trained with digits zero through five as its closed-set objects and six and seven as its open-set. Other possible arrangements are depicted in Figure 8 below for clarity. By analyzing how class configuration may affect the performance of IDOE, contingencies on specific configurations may be revealed. For example, if both the open-set examples are very visually similar like "1" and "7" this may lead to poor generalization as an open-set "8" doesn't resemble the straight lines of the open-set examples.

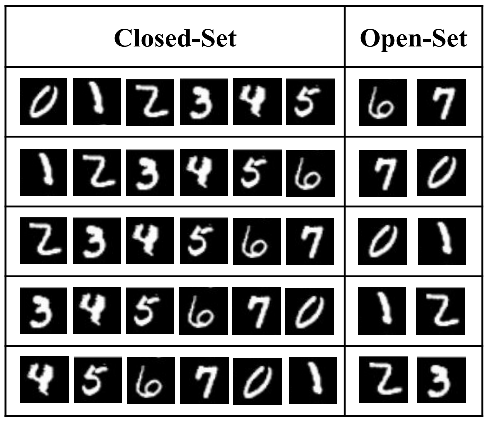


Figure 8. *Example MNIST class configurations of six closed and two open-set objects to train IDOE networks. MNIST images Source: [8]*

Besides the benchmarked MINST [8] and CIFAR-10 [10] datasets, IDOE, with ResNet-18 [15] as its base network architecture, was deployed on the PathMNIST [5] dataset of colorectal tissues to assess its applicability in a real-world clinical setting. PathMNIST was curated from a study on colorectal cancer. Of its nine classes, two are of cancerous tissues, five are of healthy tissues, and two are of inconsequential elements often encountered in the clinical setting: background (empty slides) and tissue debris, shown in Figure 9. A background and tissue debris classifying network holds minimal clinical significance, so these two classes were amalgamated to create the auxiliary open-set class.

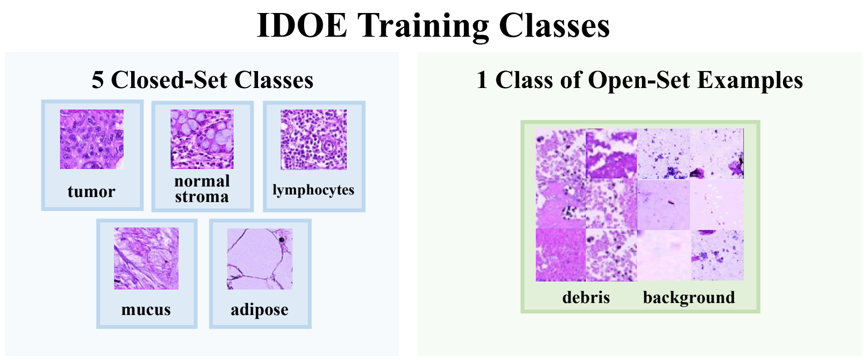


Figure 9. *When deployed on PathMNIST, IDOE has five closed-set classes and one class of open-set tissue debris and background examples. PathMNIST images Source: [5]*

5. Results

This section focuses on comparing IDOE's performance to the performance of state-of-the-art methods and deploying IDOE on the PathMNIST [5] medical dataset to evaluate its potential in preventing misdiagnosis in medical applications.

5.1 Comparison of IDOE and State-of-the-Art Methods

IDOE was used to train five networks on MNIST [8] and CIFAR-10 [10] respectively. The AUROC scores [9] of each run were recorded, averaged, and compiled in Table I below along with the mean AUROC scores of state-of-the-art methods and COE [11]. As shown, the IDOE method outperforms all other methods when applied to MNIST and all but one, KNN [7], when applied to CIFAR-10. Moreover, IDOE's AUROC is very close to KNN's AUROC on the CIFAR-10 dataset.

TABLE I. EVALUATING ALL METHODS ON MNIST AND CIFAR-10

Method	**Mean AUROC Score**	
	MNIST	***CIFAR-10***
State-of-the-Art Methods:		
OpenMax [6]	97.3%	84.2%
MSP [2]	96.2%	85.3%
ODIN [16]	98.0%	72.1%
MDS [17]	89.8%	42.9%
Gram [18]	82.3%	61.0%
EBO [19]	98.1%	84.9%
GradNorm [20]	94.5%	64.8%
ReAct [21]	82.9%	85.9%
MLS [22]	98.0%	84.8%
KLM [22]	85.4%	73.7%
VIM [23]	88.8%	83.5%
KNN [7]	97.5%	**86.9%**
DICE [24]	66.3%	79.3%
Outlier Exposure Methods:		
COE (Classic Outlier Exposure) [11]	97.1%	77.1%
IDOE (Intra-Dataset Outlier Exposure)	**98.5%**	86.3%

To further analyze whether IDOE truly outperformed other methods a t test was conducted. Since the AUROC scores of individual runs of the other state-of-the-art methods were not provided, a t test could not be performed on these methods, but a t test was performed to compare COE and IDOE. All possible configurations of six closed and two open-set objects were used to train 1260 IDOE and 210 COE networks per dataset. The resulting means, standard deviations, and p values of this analysis are shown in Table II. The p values found are less than α, so the null hypothesis can be rejected, and it can be concluded that IDOE significantly outperformed COE. Additionally, the relatively high means and low standard deviations indicate that IDOE's performance is not contingent on the closed and open-set object configurations.

TABLE II. CLASSIC VERSUS INTRA-DATASET OUTLIER EXPOSURE

Method	**MNIST**			**CIFAR-10**		
	μ	σ	*p value (α = 0.05)*	μ	σ	*p value (α = 0.05)*
COE (Classic Outlier Exposure)	97.3%	4.07%	1.89×10^{-6}	80.2%	6.65%	1.52×10^{-27}
IDOE (Intra-Dataset Outlier Exposure)	98.8%	0.98%		86.0%	4.30%	

5.2 Deploying IDOE on PathMNIST

IDOE was then evaluated on the PathMNIST [5] dataset of healthy and cancerous colorectal tissues. Since training networks on PathMNIST images required more computing power, 10 networks, each with a different configuration of closed-set objects, were trained using IDOE. However, the same auxiliary open-set class, made up of the background and tissue debris objects, were used as explained in section 4.2. The mean and standard deviation of these 10 networks' AUROC scores are shown in Table III. The relatively high mean and low standard deviation indicate that IDOE is highly effective in preventing the misdiagnosis of open-set colorectal tissues.

TABLE III. EVALUATING INTRA-DATASET OUTLIER EXPOSURE ON PATHMNIST

Method	**AUROC Data**	
	Mean	***Standard Deviation***
IDOE (Intra-Dataset Outlier Exposure)	94.8%	0.63%

In summary, IDOE consistently outperformed COE and almost all state-of-the-art methods when deployed on the MNIST and CIFAR-10 datasets. Additionally, it performed just as well on the PathMNIST dataset, demonstrating potential in similar real-world medical applications.

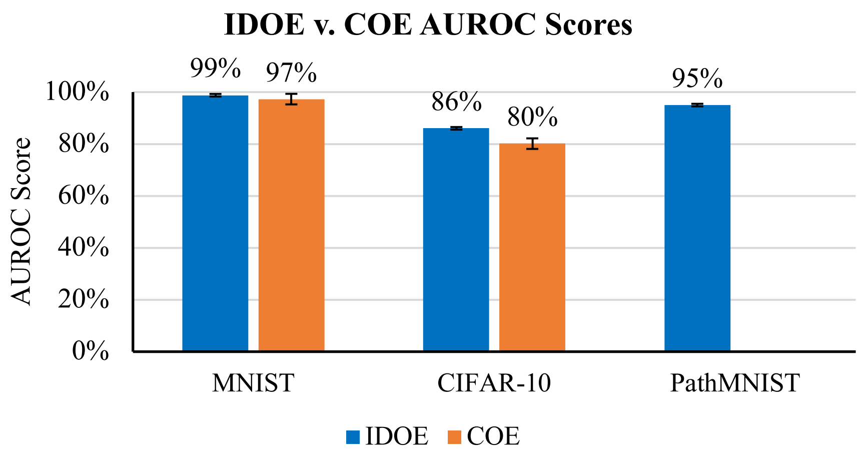


Figure 10. *IDOE consistently acheives AUROC scores above 85% across all three tested datasets.*

6. Conclusion

This study addresses the challenge of detecting open-set objects. The proposed Intra-Dataset Outlier Exposure (IDOE) method's novel auxiliary open-set class enabled networks to learn the intrinsic attributes of open-set objects. With this, the networks' understanding of open and closed-sets became flexible and adaptive. On the other hand, state-of-the-art methods rely on a more static understanding of the mathematical differences between closed and open-set objects. However, IDOE does have the drawback of requiring additional data to create the auxiliary class.

Despite this, IDOE's results are highly promising. IDOE consistently outperformed almost all state-of-the-art methods when deployed on MNIST and CIFAR-10. Furthermore, its success in recognizing outliers among colorectal tissues, even though the outliers are visually very similar to the closed-set objects, highlights its potential as a diagnostic assistant to physicians. By leveraging its ability to detect outliers, physicians can focus on diagnosing known closed set conditions. Consider a network trained with carcinoma tumors and healthy stroma as closed-set objects, and tissue debris and background in the amalgamated auxiliary open-set class. This network classifies objects not seen during training, like cancerous stroma, as open-set preventing their misclassification. Physicians can also analyze the open-set objects detected to determine if a previously undiscovered object had been encountered, furthering the potential for early detection and diagnosis of emerging medical conditions.

Acknowledgement

I would like to thank my mentor, Dr. Benjamin Erichson, a Senior Research Scientist and Research Group Leader at UC Berkeley's International Computer Science Institute, for his encouragement and advice as I completed this study and wrote this paper. He pushed me to think through every aspect of the algorithms I created to ensure my plans were thorough. Additionally, he shared his knowledge of techniques, like exponential learning rates, for robust neural network training, for which I am grateful.

Bibliography

[1] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders and K. R. Müller, "Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications," *Proc. IEEE*, vol. 109, no. 3, pp. 247-278, 2021, doi: 10.1109/JPROC.2021.3060483.

[2] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *ICLR,* 2017, doi: 10.48550/arXiv.1610.02136.

[3] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," *ArXiv preprint*, 2021, doi: 10.48550/arXiv.2110.11334.

[4] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, "Toward Open Set Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1757-1772, July 2013, doi: 10.1109/TPAMI.2012.256.

[5] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, B. Ni, "MedMNIST v2 - A large-scale lightweight benchmark for 2D and 3D biomedical image classification," *Sci Data*, vol. 10, no. 41, 2023, doi: 10.1038/s41597-022-01721-8.

[6] A. Bendale and T. E. Boult, "Towards Open Set Deep Networks," *CVPR*, Las Vegas, NV, USA, 2016, pp. 1563-1572, doi: 10.1109/CVPR.2016.173.

[7] Y. Sun, Y. Ming, X. Zhu, and Y. Li, "Out-of-distribution detection with deep nearest neighbors," *ICML*, 2022, doi: 10.48550/arXiv.2204.06507.

[8] Y. LeCun, C. Cortes, and J. C. Burges, "The MNIST Database of Handwritten Digits", 1999.

[9] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145-1159, 1997, doi: 10.1016/S0031-3203(96)00142-2.

[10] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," *Tech Report*, 2009.

[11] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep Anomaly Detection with Outlier Exposure," *ICLR*, 2019, doi: 10.48550/arXiv.1812.04606.

[12] C. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[13] J. Yang, P. Wang, D. Zou, Z. Zhou, K. Ding, W. Peng, H. Wang, G. Chen, B. Li, Y. Sun, X. Du, K. Zhou, W. Zhang, D. Hendrycks, Y. Li, Z. Liu, "OpenOOD: Benchmarking Generalized Out-of-Distribution Detection," *ArXiv preprint*, 2022, doi: 10.48550/arXiv.2210.07242.

[14] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[15] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *CVPR*, Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[16] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *ICLR*, 2018, doi: 10.48550/arXiv.1706.02690.

[17] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," *NeurIPS*, 2018, doi: 10.48550/arXiv.1807.03888.

[18] C. S. Sastry, and S. Oore, "Detecting out-of-distribution examples with gram matrices," *ICML*, 2020.

[19] W. Liu, X. Wang, J. D. Owens, and Y. Li, "Energy-based out-of-distribution detection," *NeurIPS*, 2020.

[20] R. Huang, A. Geng, and Y. Li, "On the importance of gradients for detecting distributional shifts in the wild," *NeurIPS*, 2021.

[21] Y. Sun, C. Guo, and Y. Li, "React: Out-of-distribution detection with rectified activations," *NeurIPS*, 2021.

[22] D. Hendrycks, S. Basart, M. Mazeika, M. Mostajabi, J. Steinhardt, and D. Song, "Scaling out-of-distribution detection for real-world settings," *ICML*, 2022.

[23] H. Wang, Z. Li, L. Feng, and W. Zhang, "Vim: Out-of-distribution with virtual-logit matching," *CVPR*, 2022.

[24] Y. Sun and S. Li, "Dice: Leveraging sparsification for out-of-distribution detection," *ECCV*, 2022.

[25] N. B. Erichson, S. H. Lim, W. Xu, F. Utera, Z. Cao, and M. W. Mahoney, "NoisyMix: Boosting Model Robustness to Common Corruptions," doi: abs/2202.01263.

Appendix A: IDOE AUROC Scores

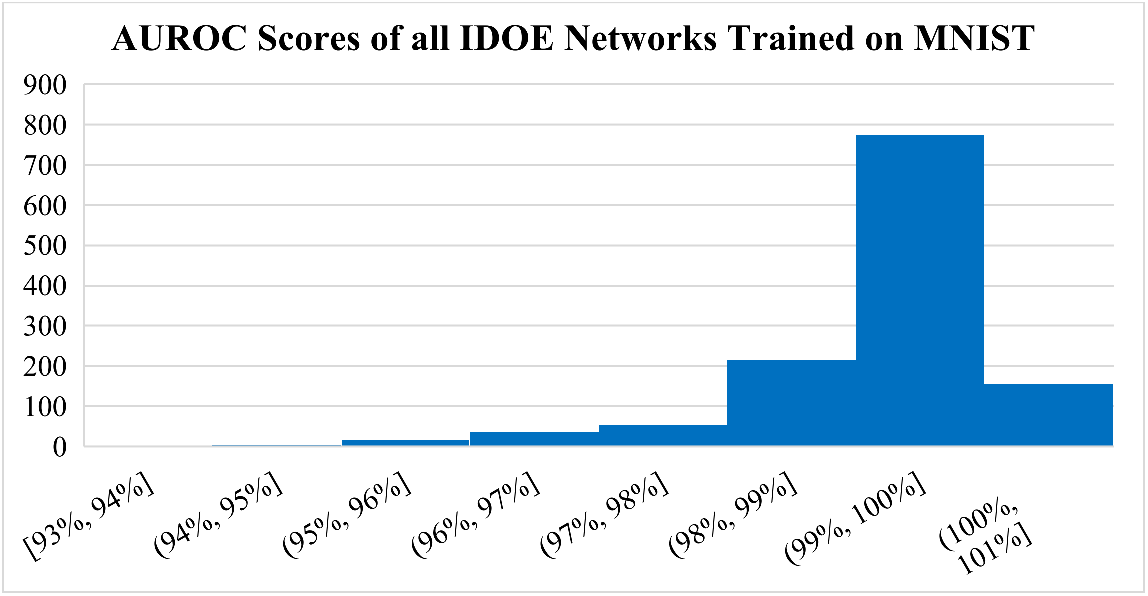
IDOE was used to train multiple networks for each dataset. Each network was trained with a different configuration of closed and open-set objects. This appendix provides detailed results of these networks.

A.1 IDOE's Implementation on the MNIST Dataset

Below are the closed-set, open-set, and AUROC scores of 10 randomly chosen networks of the 1260 total trained on MNIST.

Closed-Set						Open-Set		AUROC Score
3	4	5	6	8	9	1	2	98.86%
1	3	4	6	7	8	0	9	99.69%
0	3	4	6	7	9	2	8	99.23%
2	4	6	7	8	9	1	5	95.13%
1	2	3	5	8	9	0	6	98.46%
0	1	2	4	5	7	3	9	98.55%
0	3	4	5	6	9	7	8	98.87%
0	4	5	6	7	8	1	3	99.26%
0	3	4	6	7	8	2	9	98.81%
0	3	4	6	7	8	2	5	99.40%

Below is a histogram of the AUROC scores of the 1260 networks trained.



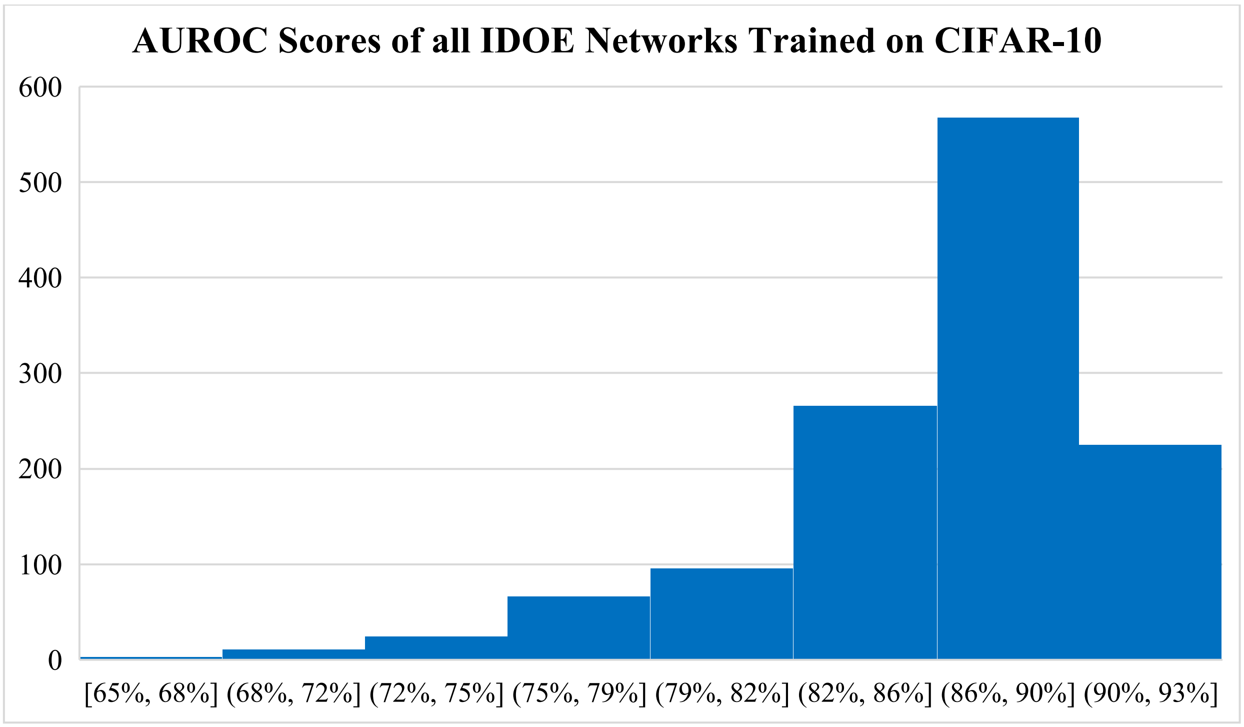
As the table and graph show, overall, IDOE consistently achieved very high AUROC scores when applied to the MNIST dataset despite changes in the configuration of its trained closed and open-set objects, indicating that its performance is not contingent on these configurations.

A.2 Sample IDOE AUROC Scores for the CIFAR-10 Dataset

Below are the closed-set, open-set, and AUROC scores of 10 randomly chosen networks of the 1260 total trained on CIFAR-10.

Closed-Set						Open-Set		AUROC Score
"deer"	"dog"	"frog"	"horse"	"ship"	"truck"	"automobile"	"bird"	83.41%
"bird"	"deer"	"frog"	"horse"	"ship"	"truck"	"automobile"	"cat"	78.23%
"bird"	"dog"	"frog"	"horse"	"ship"	"truck"	"plane"	"cat"	81.28%
"bird"	"cat"	"deer"	"frog"	"horse"	"truck"	"automobile"	"dog"	80.89%
"automobile"	"deer"	"dog"	"horse"	"ship"	"truck"	"bird"	"frog"	91.86%
"automobile"	"cat"	"dog"	"frog"	"horse"	"truck"	"plane"	"bird"	90.91%
"automobile"	"cat"	"deer"	"dog"	"horse"	"truck"	"plane"	"bird"	92.02%
"automobile"	"bird"	"dog"	"frog"	"ship"	"truck"	"plane"	"deer"	87.53%
"automobile"	"bird"	"deer"	"dog"	"horse"	"ship"	"cat"	"frog"	86.92%
"automobile"	"bird"	"cat"	"dog"	"ship"	"truck"	"plane"	"deer"	88.54%

Below is a histogram of the AUROC scores of the 1260 networks trained.



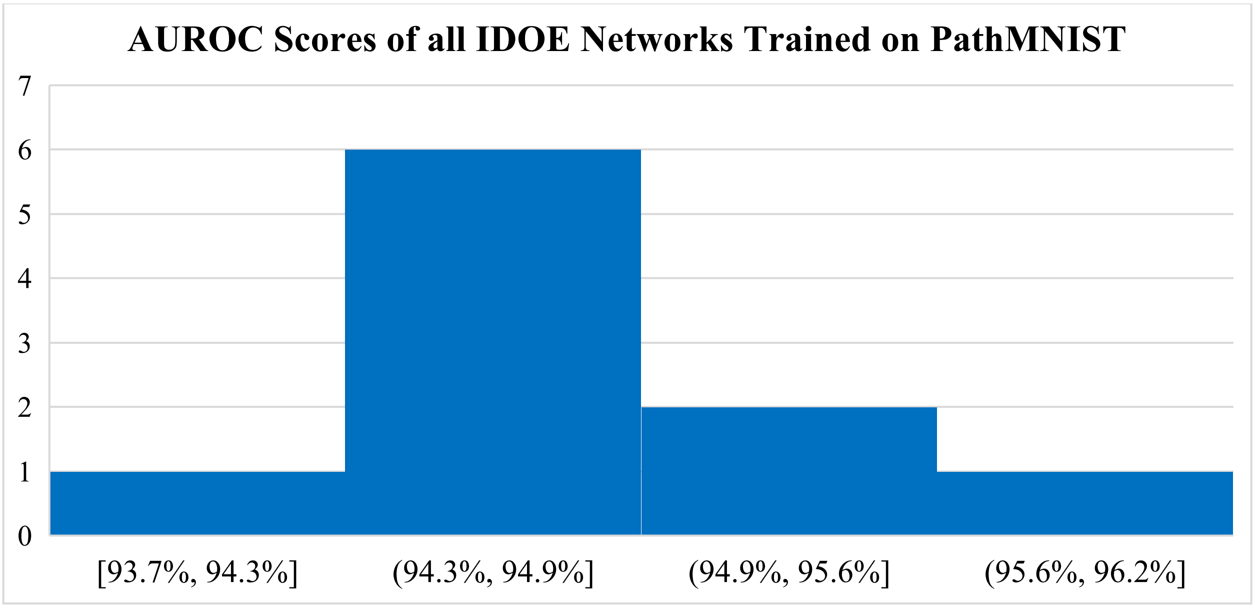
As the table and graph show, overall, IDOE consistently achieved high AUROC scores when applied to the CIFAR-10 dataset despite changes in the configuration of its trained closed and open-set objects, indicating that its performance is not contingent on these configurations.

A.3 Sample IDOE AUROC Scores for the PathMNIST Dataset

Below are the closed-set, open-set, and AUROC scores of 10 randomly chosen networks of the 11 total trained on PathMNIST.

Closed-Set					Open-Set		AUROC Score
"adipose"	"lymphocyte"	"mucus"	"cancerous stroma"	"tumor"	"background"	"debris"	94.88%
"adipose"	"lymphocyte"	"mucus"	"healthy stroma"	"tumor"	"background"	"debris"	95.02%
"adipose"	"lymphocyte"	"mucus"	"healthy stroma"	"cancerous stroma"	"background"	"debris"	94.64%
"adipose"	"lymphocyte"	"healthy stroma"	"cancerous stroma"	"tumor"	"background"	"debris"	94.42%
"adipose"	"muscle"	"healthy stroma"	"cancerous stroma"	"tumor"	"background"	"debris"	94.59%
"adipose"	"lymphocyte"	"mucus"	"muscle"	"tumor"	"background"	"debris"	93.71%
"adipose"	"lymphocyte"	"muscle"	"cancerous stroma"	"tumor"	"background"	"debris"	96.18%
"adipose"	"lymphocyte"	"mucus"	"muscle"	"cancerous stroma"	"background"	"debris"	94.40%
"adipose"	"lymphocyte"	"muscle"	"healthy stroma"	"tumor"	"background"	"debris"	94.95%
"adipose"	"mucus"	"muscle"	"cancerous stroma"	"tumor"	"background"	"debris"	93.68%

Below is a histogram of the AUROC scores of the 11 networks trained.



As the table and graph show, overall, IDOE consistently achieved high AUROC scores when applied to the PathMNIST dataset despite changes in the configuration of its trained closed and open-set objects, indicating that its performance is not contingent on these configurations.

Appendix B: Code

Below is the main portion of the code written that implements IDOE when training networks.

```
import config
import os, sys, time, shutil
import numpy as np
import utils
import argparse

def train_NN(id_classes, seen_ood_classes, x_tr, y_tr, boost, epochs, type):
  start = time.time()
  x, y = utils.balance_classes_boost(id_classes, seen_ood_classes, x_tr, y_tr, boost)
  utils.print_count_noind('run0 - 1:', y, len(y_tr))

  y_orig = y.copy() # needed to help remember original data as we relabel
  ###assert(len(id_classes) == 2)
  for j in range(len(id_classes)):
    mask = np.isin(y_orig, id_classes[j])
    y[mask] = j
  for j in range(len(seen_ood_classes)):
    mask = np.isin(y_orig, seen_ood_classes[j])
    y[mask] = len(id_classes)#2
  utils.print_count_noind('run0 - 2:', y, len(y_tr))

  num_classes = len(id_classes)+1
  if config.debug > 5:
    print('run0 - starting training')
  hist, model = utils.setup_and_train(x, y, num_classes, epochs, type)
  end = time.time()
  print('run took', round(end-start), 's')

  return model

def count_classes(pred_arr, num):
  nums = [0]*(num) # ind is the number of id classes + 1
  for i in range(len(pred_arr)):
    nums[pred_arr[i]] += 1
  return nums

def calc_auc(orig_model, id_classes, seen_ood_classes, unseen_ood_classes, x, y):
  # things in id_classes[0] should be labelled 0
  # things in id_classes[1] should be labelled 1
  # ...
  id_x = x[id_classes[0]]
  id_y = np.zeros(len(y[id_classes[0]]))
  for ind in range(1,len(id_classes)):
    id_x = np.vstack((id_x, x[id_classes[ind]]))
    id_y = np.hstack((id_y, np.ones(len(y[id_classes[ind]]))*ind))
    #print(ind, id_classes[ind])
  ind += 1
  seen_ood_x = x[seen_ood_classes[0]]
  seen_ood_y = np.ones(len(y[seen_ood_classes[0]]))*ind
  #ood_classes = seen_ood_classes + unseen_ood_classes
  for index in range(1, len(seen_ood_classes)):
    seen_ood_x = np.vstack((seen_ood_x, x[seen_ood_classes[index]]))
    seen_ood_y = np.hstack((seen_ood_y, np.ones(len(y[seen_ood_classes[index]]))*ind))
  my_x = np.concatenate((id_x, seen_ood_x))
  my_y = np.concatenate((id_y, seen_ood_y))
  unseen_ood_x = x[unseen_ood_classes[0]]
  unseen_ood_y = np.ones(len(y[unseen_ood_classes[0]]))*ind
  for index in range(1, len(unseen_ood_classes)):
    unseen_ood_x = np.vstack((unseen_ood_x, x[unseen_ood_classes[index]]))
    unseen_ood_y = np.hstack((unseen_ood_y, np.ones(len(y[unseen_ood_classes[index]]))*
ind))
  my_x = np.concatenate((my_x, unseen_ood_x))
  my_y = np.concatenate((my_y, unseen_ood_y))
  utils.print_count_noind('old', np.concatenate(y), len(y))
  utils.print_count_noind('new', my_y, len(y))

  m = utils.tf.keras.metrics.AUC(num_thresholds=200, curve="ROC", summation_method="int
erpolation", name=None, dtype=None, thresholds=None, multi_label=False, num_labels=None
, label_weights=None, from_logits=False)
  y_pred = np.argmax(orig_model.predict( my_x, verbose = 0), axis = 1)
  m.update_state(my_y, y_pred)
  print(ind, len(id_y), len(seen_ood_y), len(unseen_ood_y))
  y_pred_id = y_pred[:len(id_y)]
  y_pred_seen = y_pred[len(id_y):len(id_y)+len(seen_ood_y)]
  y_pred_unseen = y_pred[-len(unseen_ood_y):]
  print(ind, len(y_pred_id), len(y_pred_seen), len(y_pred_unseen))
  id_nums = count_classes(y_pred_id, ind+1)
  seen_nums = count_classes(y_pred_seen, ind+1)
  unseen_nums = count_classes(y_pred_unseen, ind+1)
  id_2_id = np.sum(id_nums[0:ind])
  id_2_ood = id_nums[-1]
  seen_2_id = np.sum(seen_nums[0:ind])
  seen_2_ood = seen_nums[-1]
  unseen_2_id = np.sum(unseen_nums[0:ind])
  unseen_2_ood = unseen_nums[-1]
  print('result: id', id_classes, 'seen', seen_ood_classes, 'unseen', unseen_ood_classe
s, 'auc', m.result().numpy(), id_2_id, id_2_ood, seen_2_id, seen_2_ood, unseen_2_id, un
seen_2_ood)

########################################
# Dataset specific initialization routines
########################################
def run_med(data_flag):
  train_dataset, val_dataset, test_dataset = utils.med_get_datasets(data_flag)
  x_tr, y_tr, x_va, y_va, x_te, y_te = utils.med_get_raw_data_as_classes6(train_dataset
, val_dataset, test_dataset) # ignores te, uses va for te, and splits tr to tr and va
  return x_tr, y_tr, x_va, y_va, x_te, y_te, 'lenet'

def run_mnist():
  x_tr, y_tr, x_va, y_va, x_te, y_te = utils.mnist_get_raw_data_as_classes6() # splits
tr to tr and va
  return x_tr, y_tr, x_va, y_va, x_te, y_te, 'lenet'

def run_cifar10():
  x_tr, y_tr, x_va, y_va, x_te, y_te = utils.cifar10_get_raw_data_as_classes6() # split
s tr to tr and va
  return x_tr, y_tr, x_va, y_va, x_te, y_te, 'resnet18'

########################################
# main routine
########################################
if __name__ == '__main__':

  ########################################
  # Initialization
  ########################################
  parser = argparse.ArgumentParser(description='Open Set Recognition')
  typegroup = parser.add_mutually_exclusive_group(required = True)
  typegroup.add_argument('--med', choices = ['path', 'blood', 'tissue', 'organa'])
  typegroup.add_argument('--mnist', action = 'store_true')
  typegroup.add_argument('--cifar10', action = 'store_true')
  parser.add_argument('--boost', type = float, required = False, default = 1)
  parser.add_argument('--epochs', type = int, required = False)
  parser.add_argument('--id', nargs = '+', type = int, required = True, help = 'Integer
 list of id classes')
  parser.add_argument('--seen', nargs = '+', type = int, required = True, help = 'Integ
er list of seen ood classes')
```

```
  args = parser.parse_args()
  print(args)

  if len(list(set(args.id))) < len(args.id):
    print('id: repeated class')
    sys.exit()
  if len(list(set(args.seen))) < len(args.seen):
    print('seen: repeated class')
    sys.exit()
  if len(list(set(args.id) & set(args.seen))) > 0:
    print('common element in id and seen')
    sys.exit()

  epochs = config.epochs
  if args.epochs:
    epochs = args.epochs
  if args.med:
    data_flag = args.med + 'mnist'
    x_tr, y_tr, x_va, y_va, x_te, y_te, type = run_med(data_flag)
  if args.mnist:
    x_tr, y_tr, x_va, y_va, x_te, y_te, type = run_mnist()
  if args.cifar10:
    x_tr, y_tr, x_va, y_va, x_te, y_te, type = run_cifar10()

  if len(args.id) + len(args.seen) >= len(y_tr):
    print('no classes left over for unseen')
    sys.exit()
  ########################################
  # train NN, then calculate auc
  ########################################
  id_classes, seen_ood_classes = ((args.id), (args.seen))
  unseen_ood_classes = utils.make_unseen_ood_classes(id_classes, seen_ood_classes, len(
y_tr))
  id_name = '_'.join(str(e) for e in id_classes)
  seen_ood_name = '_'.join(str(e) for e in seen_ood_classes)
  name = id_name + '-' + seen_ood_name
  print(name)

  mod = train_NN(id_classes, seen_ood_classes, x_tr, y_tr, args.boost, epochs, type)
  calc_auc(mod, id_classes, seen_ood_classes, unseen_ood_classes, x_va, y_va)
```