Challenges in OpenSet Learning of Neural Networks

Péter Bán[1]* and László Kovács[2]

[1] Department of General Informatics, Faculty of Mechanical Engineering and Informatics, University of Miskolc, Miskolc, Hungary
bn.peter.hun@gmail.com,
[2] Department of General Informatics,Faculty of Mechanical Engineering and Informatics, University of Miskolc, Miskolc, Hungary
laszlo.kovacs@uni-miskolc.hu

Abstract. Standard neural network classifiers assume a closed-world setting, where all test-time classes are known during training. In practice, models often face samples from unseen classes, a challenge known as open-set recognition. We investigate and compare two approaches: OpenMax, which fits a Weibull distribution to penultimate-layer activations to estimate class membership, and an extra-class insertion method, which explicitly models an "unknown" class during training. We evaluate both on Tiny ImageNet, CIFAR-10, MNIST, COIL-20, and the Dry Bean dataset under open-set conditions, with selected classes withheld from training. Results show that OpenMax yields only marginal gains over a closed-set baseline, largely due to unreliable Weibull fitting and invalid feature-space assumptions. Extra-class insertion consistently outperforms OpenMax, nearly doubling accuracy on unknown classes in our experiments, making it a more practical foundation for open-set learning.

Keywords: open set learning, OpenMax, extra class insertion

1 Introduction

The traditional approach to image classification trains a model on a dataset containing images and corresponding labels. It is called closed-set recognition. This assumes that all classes the trained model may encounter are known in advance, but that is usually not true. In the real world, the environment usually changes, so the deployed model may try to classify objects not included in the training set. This can lead to misclassification, which has downsides in real-world use. Tests during classification model training also do not account for this possibility [1], [2].

This situation can occur in malware detection [3], behavioral detection [4], and medical imaging [5]. Much research has focused on open set recognition, yielding many methods with promising results. In this paper, we examine OpenMax to assess its effectiveness and seek improvements.

© The Author(s) 2026
S. Árvai-Homolya et al. (eds.), *Proceedings of the 2nd Innovation in Engineering Education Conference (IEE 2026)*, Atlantis Highlights in Social Sciences, Education and Humanities 58,
https://doi.org/10.2991/978-94-6239-823-8_4

2 Literature review

Open set recognition was formalized in 2013, and since then many studies have been conducted to find methods to handle unseen categories [1], [6]. The recognition task can be divided into four categories [1].

1. **Known known classes (KKCs)**: For example, labeled training data.
2. **Known unknown classes (KUCs)**: For example, labeled negative samples.
3. **Unknown known classes (UKCs)**: No available sample, only semantic data is available.
4. **Unknown unknown classes (UUCs)**: No information about it at all.

In closed-set recognition, only KKCs are considered, and it does not handle other cases. On the other hand, open-set recognition handles KKCs as well as UCCs. This means the model learns the classes seen during training well, but it can also recognize when a class was not present during training; as a result, it will not categorize objects into known classes.

2.1 Metrics

Samples far from the known classes used in training may be labeled in open space, which is called open space risk [1]. Following [1], open space risk can be formalized in the following way:

$$R_{\mathcal{O}}(f) = \frac{\int_{\mathcal{O}} f(x)\,dx}{\int_{S_o} f(x)\,dx}$$

where f is the recognition function. f(x) = 1 means that a KKC was detected; otherwise, 0. It is a relative measure between the open space O and the measure space S_o.
Following [1] let CTA, be the set of classes to be recognized, CTR be the set of classes used in training and CTE be the set of classes used during testing. Then the openness of the corresponding recognition task O is:

$$O = 1 - \sqrt{\frac{2\,|C_{TR}|}{|C_{TA}| + |C_{TE}|}}$$

AUROC [7]: it represents the integral (the area) under a curve that plots the relationship between the True Positive Rate (TPR) and the False Positive Rate (FPR) across every possible decision threshold The AUROC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad \text{and} \qquad \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

2.2 Open Set Learning approaches

Distance-based classifiers [1], when used in open-set scenarios, cannot handle unseen classes. For this reason, many studies have adapted these methods for open-set scenarios, since the base methods have proven to be good classifiers and provide a strong foundation. One popular approach is to build clustering algorithms, which resulted in Nearest Non-Outlier and an open-set version of Nearest Neighbor (OSNN). Another approach was SVM-based methods. Rudd et al. introduced the Extreme Value Machine, a classifier that uses extreme value theory with stronger theoretical backing than prior methods.

Generative models [1] Discriminative models use the existing data to figure out if a sample is unknown. Generative models, on the other hand, produce synthetic samples to represent unknown classes in the training data, so when the classifier decides whether a sample is known, it has a reference to work with. One approach is to train Generative Adversarial Networks (GANs) to generate synthetic samples that can be used during classifier training to represent unknown samples. Research shows that this approach can make the OSR model more robust to real unknown samples.

2.3 OpenMax

OpenMax is based on extreme value theory. It was an important step in using neural networks for open-set recognition. The article [8] discusses how techniques were previously formalized that reject samples considered unknown during testing. However, it is argued that it is simply not possible to cover all UUCs (unknown unknown classes). Therefore, a method is needed that can handle this situation [8]. For most deep neural networks, in the case of classification, the output of the last fully connected layer is passed as input to the SoftMax function, whose output is a probability distribution over N known classes. One might think that when an unknown class needs to be identified, the network would assign a low probability to all known classes, and this uncertainty could be easily exploited to filter out the unknown. However, this is not the case, as has been shown in other papers [8].

The introduced OpenMax [8] method consists of the following main algorithmic steps. Instead of the Softmax output (which is already normalized), the paper uses the preceding layer's output, which it calls the activation vector. This represents the values corresponding to each class. Each class is represented by a point called the MAV (mean activation vector), which is computed as the average of the activation vectors of correctly classified instances. The distance between the MAV and the input is calculated, and using a Weibull distribution, it can be determined whether the input belongs to one of the known categories. However, the model is difficult to calibrate well [8].

The results show that, compared to Softmax, OpenMax generally achieves better F-measure and accuracy across different parameter settings, and overall, OpenMax achieves the best results [8].

3 Efficiency test on the OpenMax method

OpenMax was important when it was introduced, and it is still used as a benchmark for recent methods. It provides a great starting point for experimenting and learning about open set recognition to develop new techniques. In the OpenMax paper [8] the authors mentioned that finding the right parameters is a challenging task. The goal of the test is to see how well the model performs and whether setting the parameters is difficult, since it limits real-world usability.

For each dataset, the base model was trained for 20 epochs using adam optimizer with a batch size of 32. We used sparse categorical cross-entropy as the loss function. The learning rate was set to 10^{-3}. We tested tail sizes from 1 to 100 and found 50 to be best. We tested decision thresholds from 0.05 to 0.5, and 0.2 performed best.

We used 5-fold stratified cross-validation over the known-class training data (StratifiedKFold, scikit-learn), preserving class proportions across folds. For each fold, a new model was initialized and trained from scratch for 20 epochs. All experiments used a fixed random seed(42), applied to both NumPy (np.random.seed) and TensorFlow ($tf.random.set_seed$) prior to model construction, and passed as $random_state$to the stratified k-fold splitter with shuffling enabled.

3.1 Datasets

The MNIST dataset is one of the most well-known datasets in machine learning and consists of 70,000 images of handwritten digits (0–9). Each image is 28×28 pixels in grayscale, making it an ideal starting point for learning image classification techniques.

Tiny ImageNet [9] contains 200 categories, each with 500 training samples. For testing, there are 50 samples per class.

The CIFAR-10 [10] dataset consists of 60,000 32×32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images.

The MNIST [11] database of handwritten digits has a training set of 60,000 examples and a test set of 10,000 examples. It is a subset of the larger MNIST dataset. The digits have been size-normalized and centered in a fixed-size image.

COIL-20 [12] is a dataset of 1,440 toy images, each of size 128×128, spanning 20 classes.

Dry Bean [13] dataset includes features like shape, size, solidity, and compactness for about 7 different types of beans. There are 13,611 samples.

- 7 classes of beans
- Feature vectors describing images
- 16 attributes
- Training set: 10,888 items
- Test set: 2,723 items

To summarize, four datasets were used to evaluate OpenMax: CIFAR-10, COIL-20, MNIST, and Tiny ImageNet. These are standard datasets that are

commonly used in other open-set learning research, as well as in other fields as benchmarks.

For all datasets, we prepared the data in the same way. To create an open-set setting, we need classes present only in the training data, not in the testing data. We also need samples from classes that are not in the training data but are in the testing data. For this reason, we arbitrarily chose classes to include in the testing dataset but exclude from the training data.

For CIFAR-10, we used only the last three classes in the testing set. For the COIL-20 dataset, we similarly used only the last three classes for testing to examine whether having more known classes relative to unknown classes makes a difference. MNIST has the same number of classes as CIFAR-10, so we used the same setting, allowing us to compare how the two datasets perform under identical conditions. Tiny ImageNet has 200 categories, and we chose 10 to include only in the test set.

For the dry bean dataset, we used only 2 classes for training, and the rest were unseen during training. For training and testing, we used an 80-20 percent split. For the splits, we used the scikit-learn library with the seed value set to 42. For the dry bean dataset, we also normalized the features between 0 and 1, as this yielded better results. We divided the image RGB values by 255 to normalize them.

Table 1 summarizes the used datasets and their parameters.

Table 1. Datasets features

Dataset	Number of samples	Training samples	Testing samples	Number of classes
MNIST	70,000	41935	6989	10
COIL-20	1,440	1152	288	20
CIFAR-10	60,000	50,000	10,000	10
Tiny ImageNet	100,000	80,000	20,000	200
Dry Bean	13,611	10,888	2,723	7

3.2 Implementation

We implemented OpenMax in Python using TensorFlow. The neural network architecture is shown in Table 2.

We implemented the method as described in the paper. After preparing the dataset for the open-set setting, we first trained a neural network to classify the images. Following the paper, we took the penultimate layer and obtained its output for all training data. For each class, we calculated the mean of these outputs, which the paper calls the mean activation vector (MAV). We then fit a Weibull distribution to the vectors that were furthest from the mean, as described in [8]. Using the fitted Weibull distribution, we extracted its parameters and computed a weight for each activation vector. This allows us to calculate the probability of a sample belonging to a known class or being rejected as unknown.

Table 2. Neural network architecture

Layer	Configuration	Output Shape
Input	-	$(32, 32, 3)$
Conv2D	32 filters, (3×3) kernel, ReLU	$(30, 30, 32)$
MaxPooling2D	(2×2) pool size	$(15, 15, 32)$
Conv2D	64 filters, (3×3) kernel, ReLU	$(13, 13, 64)$
Flatten	-	10816
Dense	64 units, ReLU	64
Dense	32 units, ReLU	32
Dense (Output)	7 units (linear)	7

3.3 Results

The results show that the model is not strong with recognizing unknown classes. It can be seen in Table 3 summarizing the results.

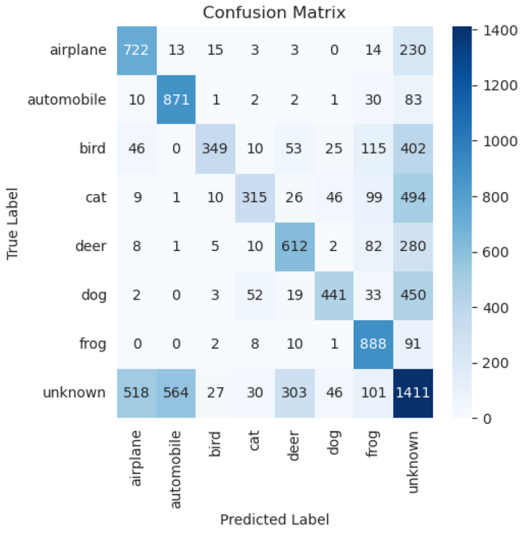


Fig. 1. Confusion matrix for the CIFAR-10 dataset

The reason for that could be that finding the right parameters is really hard as it was pointed out in the original paper [8]. Which means this method might not be very reliable. It still provided a strong foundation for using neural networks for open set learning.

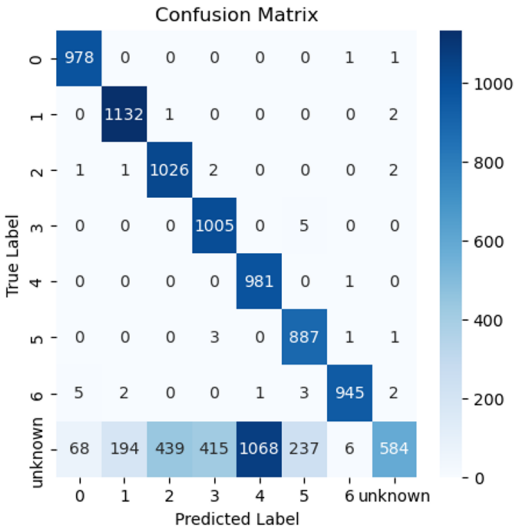


Fig. 2. Confusion matrix for the MNIST dataset

Table 3. Classification accuracy on different datasets

Dataset	Total Classes	Known Classes	All class accuracy	Accuracy with unknowns
MNIST	10	7	99.5	69.54
COIL-20	20	17	99.18	84.3
CIFAR-10	10	7	68.54	47.98
Tiny ImageNet	200	190	73.22	41.98
Dry Bean	7	2	92.76	23.9

Table 4. AUROC scores on different datasets

Dataset	Total Classes	AUROC
MNIST	10	0.8206
COIL-20	20	0.7465
CIFAR-10	10	0.3723
Tiny ImageNet	200	0.4846
Dry Bean	7	0.3790

The model can recognize known classes quite well, but for unknown's its performance is not very good. We can see in the confusion matrices shown in Figure 1 and Figure 2 that in many instances even known classes were rejected,

although the model is capable recognizing them. We also provide AUROC scores used in the literature in Table 4.

4 Extra Class Insertion Method

A closed-set classifier has K classes in its training dataset; i.e., it classifies samples into K classes. For open-set recognition, we assume samples may not belong to any of those K classes; for that reason, we apply methods like thresholding to distinguish unseen classes. The previously presented OpenMax method is weak and not applicable to real-world applications. For that reason, we also tested other methods. The most promising approach in our tests was Extra Class Insertion. This architecture applies a classifier with $K + 1$ classes; the extra class represents the unknown class. With this model the classifier tries to predict unknown samples directly into this extra class. In this approach, the network must be trained on a dataset that includes samples from this extra class. This extra data can be artificially generated [14], [15], [16].

By explicitly modeling a class for unknown or irrelevant data, this method allows models to absorb outliers and reduce the risk of misclassification. This approach is especially relevant in domains with evolving or ambiguous categories, such as waste sorting or open-world NLP [17]. The other traditional methods leave large empty spaces between them. Based on our experiments, this assumption is not a valid solution in most cases, as the classifiers stretch the sampled objects quasi-uniformly in the classification space.

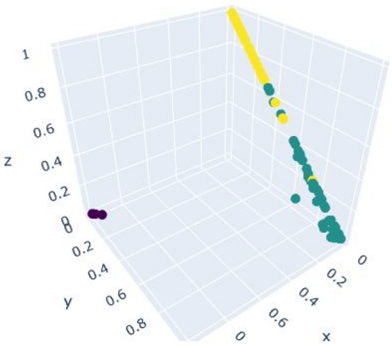


Fig. 3. Object space in the output layer of the Iris dataset

As shown in Figure 3, backpropagation stretches the objects to cover the entire object space and better separates the classes. These experiments show

that all objects from untrained categories map to positions near the existing class regions. The only way to ensure a significant separation is to apply training items from a different, distinct category. Expanding the training datasets with strange, extra items shrinks the existing category areas and creates a larger separation space between these zones.

The main critical factors of the extra -class insertion method can be summarized as follows.

- Clutter, occlusion, and class imbalance remain challenging [18].
- Real-world waste settings are less studied than controlled datasets [19].
- High additional computational costs [20].

We have tested this approach on three benchmark datasests (Table 5).

Table 5. Benchmark dataset parameters

Dataset	Known classes	total classes	Train size	Reduced train size	Test size
MNIST	5	10	10000	2000	10000
CIFAR-10	5	10	5000	514	100
Dry Bean	2	7	10888	1481	2723

In the experimental part, we tested the following algorithm variants:

- A: All classes are known in both the train and test sets; no unknown class;
- B: Only the selected classes are used in both the train and test, no unknown class;
- C: Only the selected classes are known in the train, while the test involves all classes (including unknown classes);
- D: Application of the OpenMax method, only the selected classes are known in the train, while the test involves all classes (including unknown classes);
- E: Application of the proposed Extra Class method, only the selected classes are known in the train, while the test involves all classes (including unknown classes);

In the Extra Class variant, we generate a random training dataset in the input object feature space. In our tests, we applied a uniform random distribution. All items in this extra dataset D_E belong to the extra class. During training, we used both datasets (D_E and the usual D_B with known class memberships) in an alternating sequence ($D_E, D_B, D_E, D_B \ldots$). In the implementation, we had to adjust the following three key parameters (Table 6):

- Size of D_E
- Epoch number
- Learning rate

High learning rate caused efficiency degradation, as the engine remembered only the last dataset; it forgot either D_E or D_B.

Table 6. Parameter settings of the methods

Dataset	Method	learning rate	epochs (outer/inner)	size of D_E
MNIST	A-D	1e-3	25	
MNIST	E	1e-4	25/1	1040
CIFAR-10	A-D	1e-3	25/	
CIFAR-10	E	1e-3	25/1	514
Dry Bean	A-D	1e-3	100/	
Dry Bean	E	1e-4	100/1	10000

Table 7. Accuracy results of the methods

Dataset	acc A	acc B	acc C	acc D	acc E
MNIST	98.4	99.5	51.5	81.5	87.4
CIFAR-10	60.2	69.5	31.4	32.7	42.5
Dry Bean	92.5	100	13.9	14.8	90.7

In our tests, we found the following values to be optimal for the parameters.

The measured accuracy values are summarized in Table 7.

Thus, based on our experience, the Extra Class insertion method is a promising candidate for implementing a toolset for open-set learning. The main goal of our further investigations is to analyze the method's efficiency across different application cases.

Conclusion

In practical applications, neural network models often face samples from unseen classes, a challenge known as open-set recognition. We investigate and compare two approaches: the benchmark OpenMax, which fits a Weibull distribution to penultimate-layer activations to estimate class membership, and a proposed Extra-Class insertion method, which explicitly models an "unknown" class during training.

Our experiments show that OpenMax method struggle to reliably recognize unknown classes. This is likely due to the difficulty of correctly fitting the Weibull distribution and tuning its parameters, a limitation also noted in the original OpenMax paper [8]. While the model classifies known classes well, it also frequently rejects them, as shown in the confusion matrices. The Extra-Class insertion method, which explicitly models an "unknown" class, achieved a great and consistent improvement. The results show much higher accuracy and easier configuration. For these reasons, this approach is well-suited for practical open-set learning.

The main contributions of the paper can be summarized as follows:

- Proposal a novel Extra-Class method for the OpenSet Recognition task.
- The proposed Extra-Class method dominated the widely use baseline method OpenMax in all test cases.

– The analysis performed shows the difficulties facing us in the OpenSet problem domain.

The proposed method is a promising candidate for implementing an efficient toolset for open-set learning. As our tests covered only a few benchmark datasets, more research work is needed to provide a stable, general efficient solution. Some open topics remain for further investigation, including handling high-dimensional object space and finding an optimized distribution of the extra data items.

References

1. Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. "Recent advances in open set recognition: A survey". In: *IEEE transactions on pattern analysis and machine intelligence* 43.10 (2020), pp. 3614–3631.
2. Marcos Barcina-Blanco et al. "Managing the unknown: a survey on open set recognition and tangential areas". In: *arXiv preprint arXiv:2312.08785* (2023).
3. Xue Li et al. "Open set recognition for malware traffic via predictive uncertainty". In: *Electronics* 12.2 (2023), p. 323.
4. Alina Roitberg et al. "Open set driver activity recognition". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2020, pp. 1048–1053.
5. Yiqian Xu et al. "Semi-supervised and class-imbalanced open set medical image recognition". In: *IEEE access* 12 (2024), pp. 122852–122877.
6. Sagar Vaze et al. "Open-set recognition: A good closed-set classifier is all you need". In: *International conference on learning representations*. 2022.
7. Suraj Regmi. *Mathematical Interpretation of AUROC*. 2024. URL: https://surajregmi.medium.com/mathematical-interpretation-of-auroc-9fc8d4f43ce3 (visited on 07/20/2026).
8. Abhijit Bendale and Terrance E Boult. "Towards open set deep networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1563–1572.
9. Ya Le and Xuan Yang. "Tiny imagenet visual recognition challenge". In: *CS 231N* 7.7 (2015), p. 3.
10. Alex Krizhevsky and Geoffrey Hinton. "Learning multiple layers of features from tiny images". In: (2009).
11. Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
12. S.K. Nayar S.A. Nene and H. Murase. "Columbia Object Image Library (COIL-20)". In: *Technical Report, Department of Computer Science, Columbia University CUCS-005-96*. Feb. 1996.
13. *Dry Bean*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C50S4B. 2020.
14. ZongYuan Ge et al. "Generative OpenMax for Multi-Class Open Set Classification". In: *CoRR* abs/1707.07418 (2017). arXiv: 1707.07418. URL: http://arxiv.org/abs/1707.07418.

15. Wonwoo Cho and Jaegul Choo. *Towards Accurate Open-Set Recognition via Background-Class Regularization*. 2022. arXiv: 2207.10287 [cs.CV]. URL: https://arxiv.org/abs/2207.10287.
16. Akshay Raj Dhamija, Manuel Günther, and Terrance Boult. "Reducing network agnostophobia". In: *Advances in Neural Information Processing Systems* 31 (2018).
17. Suman Kunwar. "The garbage dataset (GD): A multi-class image benchmark for automated waste segregation". In: *arXiv preprint arXiv:2602.10500* (2026).
18. Tachin Chin et al. "Artificial intelligence-related technologies for municipal solid waste management: A review and new insights from smart cities". In: *Sustainable Cities and Society* (2026), p. 107530.
19. Bouchra Termass et al. "CNNs vs. transformers: A benchmark for multi-class marine debris identification". In: *Science of The Total Environment* 1039 (2026), p. 181902.
20. Zhiming Dong et al. "Benchmarking computer vision models for automated construction waste sorting". In: *Resources, Conservation and Recycling* 213 (2025), p. 108026.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (http://creativecommons.org/licenses/by-nc/4.0/), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

