MMF: A loss extension for feature learning in open set recognition*

Jingyun Jia[0000-0003-0865-049X] and Philip K. Chan[0000-0002-3878-4205]

Florida Institute of Technology, Melbourne FL 32901, USA
`jiaj2018@my.fit.edu pkc@fit.edu`

Abstract. The objective of open set recognition (OSR) is to classify the known classes as well as the unknown classes when the collected samples cannot exhaust all the classes. This paper proposes a loss extension that emphasizes features with larger and smaller magnitudes to find representations that can more effectively separate the known from the unknown classes. Our contributions include: First, we introduce an extension that can be incorporated into different loss functions to find more discriminative representations. Second, we show that the proposed extension can significantly improve the performances of two different types of loss functions on datasets from two different domains. Third, we show that with the proposed extension, one loss function outperforms the others in training time and model accuracy.

Keywords: Open set recognition · Feature learning · Loss extensions.

1 Introduction

The OSR problem aims to classify the multiple known classes for a multinomial classification problem while identifying the unknown classes. The OSR problem defines a more realistic scenario and has drawn significant attention in application areas such as face recognition [12], malware classification [5] and medical diagnoses [15].

In this paper, we introduce a loss extension to help the existing loss functions better handle the open set scenario. The proposed extension is inspired by Extreme Value Signatures (EVS) in [17]. Borrowing from a pre-trained neural network for regular classification, EVS uses only the top K activations (i.e., largest in magnitude) at one layer for calculating the distance between an instance and a class. The EVS distance function can help identify the unknown class. Instead of using a pre-trained network and the top K activations, we directly emphasize features with the largest, as well as smallest, magnitudes during network training. We name our approach Min Max Feature (MMF). Although the MMF extension is not a standalone loss function, it can be incorporated into different loss functions. Our contribution in this paper is threefold: First, we propose MMF as an extension to different types of loss functions for the OSR

* Partially supported by grants from Amazon and Rockwell Collins to Philip Chan.

problem. Second, we show that MMF achieves statistically significant AUC ROC improvement when applied to two types of loss functions (classification and representation loss functions) on four datasets from two different domains (images and malware). Third, our results indicate that the combination of MMF and the ii loss function [5] outperforms the other combinations in both training time and overall F1 score.

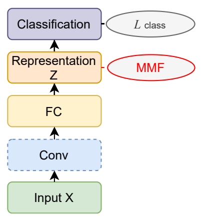
We organize the paper as follows. In section 2, we give an overview of related work. Section 3 presents the MMF loss extension. Finally, section 4 shows that the MMF extension can improve different types of loss functions significantly.

2 Related Work

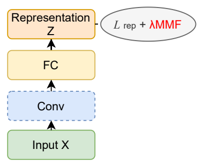
The OSR problem is related to PU (Positive and Unlabeled) learning [10], which can be regarded as a binary classification problem with the absence of negative samples. The OSR problem extends the binary classification problem to a multi-class classification problem, with some classes missing from the training set, and will be recognized as an unknown class during testing. We can divide OSR techniques into three categories based on the training set compositions. The first category includes the techniques that borrow additional data in the training set. Dhamija et al. [2] utilize the differences of feature magnitudes between known and borrowed unknown samples as part of the objective function. Hendrycks et al. [6] propose Outlier Exposure(OE) to distinguish between anomalous (unknown) and in-distribution (known) examples. In general, although borrowing and annotating additional data turns OSR into a common classification problem, the retrieval and selection of additional datasets remain an issue.

The research works that generate additional training data fall in the second category of open set recognition techniques. Most data generation methods are based on GANs. Neal et al. [11] add another encoder network to traditional GANs to map from images to a latent space. Lee et al. [9] generate "boundary" samples in the low-density area of in-distribution acting as unknown samples. While generating unknown samples for the OSR problem has achieved great performance, it requires more complex network architectures.

The third category of open set recognition does not require additional data. Most of the research works require outlier detection for the unknown class. Pidhorskyi et al. [13] propose manifold learning based on training an Adversarial Autoencoder (AAE) to capture the underlying structure of the distributions of known classes. Hassen and Chan [5] propose ii loss for open set recognition. It first finds the representations for the known classes during training and then recognizes an instance as unknown if it does not belong to any known classes. In EVS, Schultheiss et al. [17] investigate class-specific representations for novelty detection tasks. The research work shows that each class's mean representation can capture discriminative information of both known and unknown classes. EVS focuses on the top K activations via binarizing the activations; however, choosing an appropriate K can be challenging. Also, EVS assumes that all the



(a) With classification loss



(b) With representation loss

Fig. 1: An overview of the network architectures of different types of loss functions. The convolutional layers are optional. The MMF module in red is our proposed loss extension.

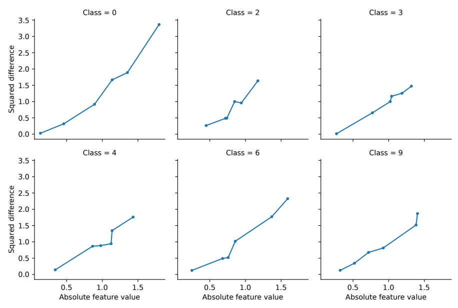


Fig. 2: Squared differences of MAV values between the known and unknown classes in Figure 3a. The x-axis is the absolute feature values in six features, and the y-axis is their corresponding squared differences to the unknown class.

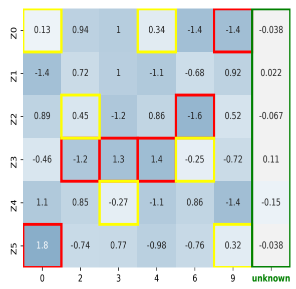
activation values are positive and only looks at the larger ones. We address both limitations in our proposed approach.

While our approach can be incorporated into different loss functions, we focus on two types of loss functions in this paper: the classification loss functions and the representation loss functions. The objective of classification loss, such as cross-entropy loss, is to lower the classification error of the training data. The representation loss functions are normally applied to the representation layers, such as triplet loss in [16] and ii loss in [5]. Triplet loss intends to find an embedding space where the distance between an anchor instance and another instance from the same class is smaller by a user-specified margin than the distance between the anchor instance and another instance from a different class. Ii loss aims to maximize the distance between different classes (inter-class separation) and minimize the distance of an instance from its class mean (intra-class spread).

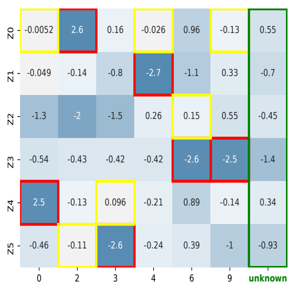
3 Approach

We propose the MMF extension to learn more discriminative representations through known classes, thus better separating known and unknown classes. The proposed MMF extension does not borrow or generate additional data for the unknown class, and it can be incorporated into different loss functions. We focus on classification loss functions such as cross-entropy loss and representation loss functions, such as triplet loss and ii loss (Section 2).

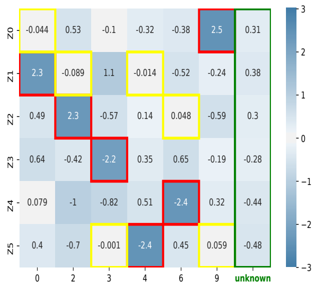
A typical classification neural network consists of an input layer, hidden layers, and classification layer. We can consider the hidden layers as different levels of representations of the input. We call the values of the last hidden layer activation vector (AV), and each activation is a learned feature. The mean activation vectors (MAV) of a class is the average of the activation vectors of the class.



(a) Standalone ce



(b) With MaxF



(c) With MMF

Fig. 3: The heatmap of MAVs (columns) of the MNIST classes using cross-entropy loss without and with different extensions. Each row is a learned feature. The largest/smallest magnitude magnitude of a feature in each MAV is in a red/yellow box. MAV of the unknown class is in a green column/box.

For example, the network in Figure 1a contains one convolutional layer, one fully connected layer, one representation layer (representation layer Z), and one classification layer (softmax layer). In some scenarios, a neural network only consists of the input layer and hidden layers as in Figure 1b, where we use learned representations instead of a classification layer for classification tasks. Figure 3a shows the learned MAV values from the representation layer using standalone cross-entropy loss.

To improve the accuracy of detecting open set samples from unknown classes, we can increase the distances (we use Euclidean distance here) between the learned features of known and unknown samples, summarized by the MAVs of the known and unknown classes. Squared differences are the components of Euclidean distance. Thus we can increase the distance by increasing squared differences. Figure 2 depicts the relationship between squared differences with the absolute feature values (feature magnitudes) of the six known classes. We consider a feature with a larger magnitude is more significant than that with a smaller magnitude. We observe that a more significant feature leads to a higher squared difference to the unknown class. The reason is that the MAV of the unknown class has a relatively small magnitude (green column), as we observe in Figure 3a. The small magnitude is due to the unknown class being absent from training, and hence its features are not learned. More importantly, the squared difference increases faster with more significant features, which indicates a slight improvement in a more significant feature will increase squared difference more. Thus, we want the features with larger magnitudes to become even more significant to increase the distance between the unknown and known classes.

However, based on the preliminary experiments, we found that after enlarging the magnitudes of the most significant features for the known classes, the unknown class's MAV became further away from the origin, which reduces the increase in the distance between the known and unknown classes. As shown in

Figure 3b, the MAV of the unknown class (green column) has significantly increased compared to the one only using standalone cross-entropy loss in Figure 3a. To further improve accuracy and increase the magnitudes of the most significant feature, we also decrease the magnitudes of the least significant features to mitigate the increase of the MAV of the unknown class. Comparing Figure 3c and Figure 3a, we can see that after reducing the magnitude of the least significant features, the feature values of unknown classes indeed get smaller. Consequently, the distance between the MAV of a known class and the MAV of the unknown class has increased, and the classes are more separated. For example, the Euclidean distance between class "9" and the unknown class learned from standalone cross-entropy loss in Figure 3a is 2.32. After adding "MMF" in 3c enlarges the distance to 2.62, making the two classes more separable.

Therefore, our MMF extension has two properties. Property A maximizes the most significant feature, i.e., the feature with the largest magnitude, for all the known classes. Property B minimizes the least significant feature, i.e., the feature with the smallest magnitude, for all the known classes. As a result, the learned representations for known classes should be more discriminative, while the unknown classes should be less affected.

3.1 Learning objectives

Let $\boldsymbol{x} \in \mathbf{X}$ be an instance and $y \in Y$ be its label. The hidden layers in a neural network can be considered as different levels of representations of input $\boldsymbol{x}$. Suppose that there are C known classes in training data, and $C+1$ classes in test data with the additional class as unknown class. We denote the MAV of class i as μ_i, and μ_{ij} represents the j^{th} feature of the MAV of class i. Assume the AVs and MAVs have F dimensions, representing F features, we stack the MAVs for all the classes to form a representation matrix $\mathbb{U}^{C\times F}$. To satisfy Property A, we first select the most significant features for each class to form the "max_feature" vector. The i^{th} element in "max_feature" is for class i:

$$max\_feature_i = \max_{1\le j\le F} |\mu_{ij}|, \tag{1}$$

In the example of Figure 3a, the "max_feature" would be (1.8, 1.2, 1.3, 1.4, 1.6, 1.4) (the absolute values of the red boxes). Likewise, for Property B, we measure the vector of the "min_feature" as the least significant feature for each class. The i^{th} element is for class i:

$$min\_feature_i = \min_{1\le j\le F} |\mu_{ij}| \tag{2}$$

The "min_feature" in the example of Figure 3a would be (0.13, 0.45, 0.27, 0.34, 0.25, 0.32) (the absolute values of the yellow boxes). Then, to maximize all the values in the "max_feature", we maximize the lower boundary (i.e., the smallest value) in "max_feature" directly. Thus the most significant features for all the known classes would be maximized as Property A. Meanwhile, we minimize the largest value in the "min_feature" to implicitly minimize all the values in the "min_feature". The least significant features for all the known classes would be minimized as Property B. As a result, the proposed MMF extension satisfies both properties:

$$MMF = \max_{1 \leq i \leq C}(min\_feature_i) - \min_{1 \leq i \leq C}(max\_feature_i) \tag{3}$$

In the example of Figure 3a, we would like to maximize the "1.2" in the "max_feature" and minimize the "0.45" in the "min_feature". There are alternative methods to generate the "max_feature" and "min_feature", for example, instead of selecting the highest absolute values for "max_feature", we experimented with the highest values ($max\_feature_{1i} = \max_{1 \leq j \leq F}(\mu_{ij})$) and the lowest values ($max\_feature_{2i} = \max_{1 \leq j \leq F}(-\mu_{ij})$) to form two "max_feature" vectors and later to be maximized at the same time. However, our experiments indicate that using the single "max_feauture" vector can achieve better performances. There are also other methods to implicitly maximize the most significant features and minimize the least significant values for all the classes, such as maximizing the average value of the "max_feature", or minimizing the average value of the "min_feature", i.e. $\sum_{i=1}^{C} \frac{1}{C}(min\_feature_i - max\_feature_i)$. However, the results of using average value are weaker than using the extreme values across all classes, hence we choose to use the extreme values in our extension function and in our experiments.

3.2 Training with MMF and Open Set Recognition

In addition to Properties A and B, the MMF extension can be incorporated into different loss functions. We focus on two types of loss functions: a) loss functions designed for decision layers such as cross-entropy loss; b) loss functions designed for representation layers such as triplet loss and ii loss. Notably, we combine the MMF extension with these two types of loss functions differently, as Figure 1.

We use the network architecture in Figure 1a to simultaneously train the network with classification loss functions and the MMF extension. During each iteration, first, we extract AVs and generate the representation matrix; second, we construct the MMF extension function from the "max_feature" vector and "min_feature" vector; third, the weights of each layer of the network are first updated to minimize the MMF extension then updated to minimize classification loss functions using stochastic gradient descent.

The MMF extension can also be incorporated into representation loss functions such as triplet loss and ii loss. As both representation loss functions and the MMF extension should be applied to the layer learning representations, their combination gives us:

$$\mathcal{L} = \mathcal{L}_{rep} + \lambda MMF, \tag{4}$$

$\mathcal{L}_{rep}$ is a representation loss function, and λ is a hyperparameter that strikes a balance between the representation loss function and the MMF extension. Figure 1b shows the network architecture using a representation loss function with an MMF extension. The combination serves on the Z-layer of the network. Moreover, the network weights get updated using stochastic gradient descent during each iteration.

After the training process, we obtain the representation centroids for each class. Then during the inference, we use the same strategy as used in ii loss [5]. First, we calculate the outlier score as the distance of learned representation to the nearest representation centroid. Then we sort the outlier score of the training data in descending order and pick the 99 percentile outlier score value as the outlier threshold. If the outlier score of a test sample exceeds the threshold, it will be recognized as the unknown class. Otherwise, it will be classified as the known class with the nearest representation centroid.

4 Experimental Evaluation

We evaluate the MMF extension with simulated open-set datasets from the following four datasets.
MNIST [14] contains 70,000 handwritten digits from 0 to 9, which is 10 classes in total. To simulate an open-set dataset, we randomly pick six digits as the known classes participant in the training, while the rest are treated as the unknown class only existing in the test set.
CIFAR-10 [7] contains 60,000 32x32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. We first convert the color images to grayscale and randomly pick six classes out of the ten classes as the known classes, while the remaining classes are treated as the known class only existing in the test set.
Microsoft Challenge (MC) [8] contains disassembled malware samples from 9 families. We use 10260 samples that can be correctly parsed then extract their function call graphs (FCG) as in [4] for the experiment. The dimensionality of the FCG is 63x63. Again, to simulate an open-set dataset, we randomly pick six classes as the known classes, while the rest are considered unknowns.
Android Genome (AG) [18] consists of malicious android apps from many families in different sizes. We use nine families (986 samples) with a relatively larger size for the experiment to be fairly split into the training set, the test set, and the validation set. we first use [3] to extract the function instructions and then extract 1453 raw FCG features as in [4]. Like the MNIST and the MC dataset, we randomly pick six classes as the known classes in the training set and consider the rest as the unknown class, which are only used in the test phase.

4.1 Network Architectures and Evaluation Criteria

We evaluate the MMF extension associated with two types of loss functions: classification loss functions and representation loss functions. Specifically, we use the cross-entropy loss as the example of classification loss functions, and use ii loss [5] and triplet loss [16] as the examples of representation loss functions. Moreover, we compare these pairs with OpenMax [1].

For the MNIST dataset, the padded input layer is of size (32, 32), followed by two non-linear convolutional layers with 32 and 64 nodes. We also use the max-pooling layers with kernel size (3, 3) and strides (2, 2) after each convolutional layer. We use two fully connected non-linear layers with 256 and 128 hidden units after the convolutional component. Furthermore, the linear layer Z, where we extract the representation matrix, is six dimensions in our experiment. We use the Relu activation function for all the non-linear layers and set the Dropout's keep probability as 0.2 for the fully connected layers. We use Adam optimizer with a learning rate of 0.001. The network architecture of the CIFAR-10 experiment is similar to the MNIST dataset, except the padded input layer is of size (36, 36). The experiment for the MS Challenge dataset also implements two convolutional layers. The padded input layer is of size (67, 67). However, we only use one fully connected layer instead of two after the convolutional layers. Also, we make the keep probability of Dropout as 0.9. The Android Genome dataset does not use the convolutional component. We use a network with one fully connected layer of 64 units before the linear layer Z. We also used Dropout with a keep probability of 0.9 for the fully connected layers. We set the learning rate of Adam optimizer as 0.1. Besides, we use batch normalization in all the layers to prevent features from getting excessively large. And as mentioned in section 3.2, we use contamination ratio of 0.01 for the threshold selection.

As we discussed in Equation 4, we use a hyperparameter λ combine the MMF extension with the representation loss functions (i.e. ii loss and triplet loss in the experiments) as: $\mathcal{L} = \mathcal{L}_{rep} + \lambda MMF$. While the range of λ is (0, 1], we set λ as 0.2 and 0.5 for ii loss and triplet loss for the MNIST and CIFAR-10 datasets. For the MC dataset, we set λ as 0.5 and 0.3 for ii loss and triplet loss. We set λ as 0.4 for both ii loss and triplet loss in the AG dataset's experiments.

We simulate three different groups of open sets for each dataset then repeat each group 10 runs, so each dataset has 30 runs in total. When measuring the model performance, we use the average AUC scores under 10% and 100% FPR (False Positive Rate) for recognizing the unknown class, as lower FPR is desirable in the real world for cases like malware detection. Furthermore, we measure the F1 scores for known and unknown classes ($C+1$ classes) separately as one of the OSR tasks is to classify the known classes. Moreover, we perform t-tests with 95% confidence in both the AUC scores and F1 scores to see if the proposed MMF extension can significantly improve different loss functions.

4.2 Experimental Results

We compare the model performances of OpenMax as well as three loss function quadruples: cross-entropy loss, ii loss, and triplet loss. Table 1 shows the AUC scores of the models in the four datasets; mainly, we focus on comparing the "Standalone" with the corresponding "+MMF" subcolumns. We observe that the quadruples, in general, achieve better AUC scores than OpenMax. Moreover, with the MMF extension, the AUC scores of the loss functions have achieved statistically significant improvements in 16 out of 24 cases (3 loss functions$\times$4 datasets$\times$2 FPR values).

Table 1: The average AUC scores of 30 runs at 100% and 10% FPR of OpenMax and three loss functions quadruples. The underlined values are statistical significant better than the standalone loss functions via t-test with 95% confidence. The values in bold are the highest values in each quadruple. The values in brackets are the highest values in each row.

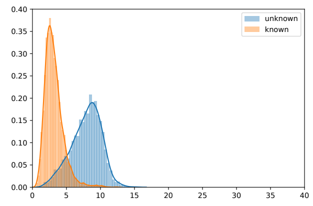
		OpenMax	ce				ii				triplet			
	FPR		Standalone	+MMF	+MaxF	+MinF	Standalone	+MMF	+MaxF	+MinF	Standalone	+MMF	+MaxF	+MinF
MNIST	100%	0.9138	0.9255	0.9479	**0.9515**	0.9393	0.9578	[**0.9649**]	0.9579	0.9607	0.9496	**0.9585**	0.9480	0.9404
	10%	0.0590	**0.0765**	0.0744	0.0761	0.0751	0.0821	[**0.0842**]	0.0826	0.0830	0.0750	**0.0796**	0.0777	0.0739
CIFAR-10	100%	[0.6757]	0.5803	0.5982	**0.6103**	0.5807	0.6392	0.6419	0.6437	**0.6439**	0.6106	**0.6248**	0.6131	0.6127
	10%	0.0065	0.0070	0.0089	**0.0090**	0.0077	[**0.0103**]	0.0096	0.0100	0.0100	0.0089	**0.0102**	0.0092	0.0093
MC	100%	0.8739	0.9148	[**0.9500**]	0.9387	0.9352	0.9385	**0.9461**	0.9407	0.9397	0.9240	**0.9430**	0.9317	0.9178
	10%	0.0405	0.0530	**0.0635**	0.0600	0.0588	0.0627	[**0.0656**]	0.0629	0.0619	0.0565	**0.0622**	0.0563	0.0546
AG	100%	0.4150	0.7506	**0.8205**	0.8152	0.7501	0.8427	0.8694	0.8763	[**0.8831**]	0.8271	**0.8379**	0.8203	0.8256
	10%	0.0010	0.0058	0.0148	**0.0163**	0.0036	0.0285	0.0305	[**0.0368**]	0.0366	0.0229	**0.0275**	0.0260	0.0235

Table 2: The average F1 scores of 30 runs of OpenMax and three loss functions pairs. The underlined values show statistically significant improvements (t-test with 95% confidence) comparing with the standalone loss functions. The values in bold are the highest values in each column.

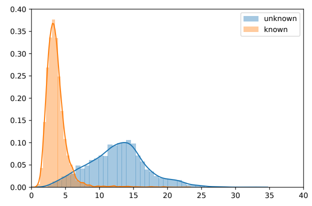
		MNIST			CIFAR-10			MC			AG		
		Known	Unknown	Overall	Known	Unknown	Overall	Known	Unknown	Overall	Known	Unknown	Overall
OpenMax		0.8964	**0.7910**	0.8814	**0.6456**	**0.5407**	**0.6306**	0.8903	0.7329	0.8679	0.2273	**0.7761**	0.3057
ce	Standalone	0.7596	0.7561	0.7591	0.5672	0.3697	0.5390	0.8881	0.6643	0.8562	0.5346	0.0033	0.4587
	+MMF	0.8504	0.7902	0.8809	0.5994	0.3271	0.5605	0.9090	**0.7963**	0.8929	0.5555	0.1142	0.4925
ii	Standalone	0.9320	0.8833	0.9250	0.6206	0.3570	0.5829	0.9128	0.7257	0.886	0.6349	0.6677	0.6396
	+MMF	**0.9373**	0.8916	**0.9308**	0.6205	0.3660	0.5842	**0.9210**	0.7680	**0.8991**	**0.6407**	0.7251	**0.6528**
triplet	Standalone	0.9103	0.8302	0.8989	0.5798	0.4515	0.5614	0.8998	0.7018	0.8715	0.5929	0.6323	0.5986
	+MMF	0.9239	0.8625	0.9152	0.5943	0.4790	0.5778	0.9064	0.7213	0.8800	0.6005	0.6895	0.6132

Table 2 shows the average F1 scores for the four datasets. We first calculate the F1 scores for each of the C known classes and the unknown class, then average the $C + 1$ classes as the Overall F1 scores. We can see that the loss functions with the MMF extension have better results than their corresponding standalone versions for both the known and the unknown classes. We observe that ii loss with the MMF extension is more accurate than the other five methods in six out of twelve F1 scores. Particularly, it achieves the highest Overall F1 scores for three out of four datasets.

Table 3 shows the comparison of the average training time of the 30 runs for the MNIST dataset with 5000 iterations via NVIDIA Tesla K80 GPU on AWS. We find that adding the MMF extension almost doubles the training time of using standalone cross-entropy. While for ii loss and triplet loss, adding the extension increases the training time by around 1%. The reason is that the MMF extension needs to create the representation matrix from scratch for the network with ce loss, which needs an extra backpropagation step, both of which take more time. We also observe that ii loss has the fastest training time among three loss functions with our MMF extension. Overall F1 scores and training time indicate that “ii+MMF” is the most accurate and efficient combination.



(a) triplet



(b) triplet+MMF

Fig. 4: The distributions of outlier scores in MNIST.

Table 3: The comparison of training time for the MNIST dataset.

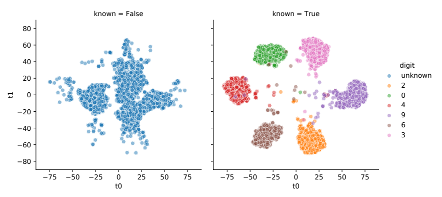
	Regular	+MMF	delta
ce	119.33	230.43	+111.1
ii	122.17	123.30	+1.14
triplet	223.27	225.70	+2.43

4.3 Analysis

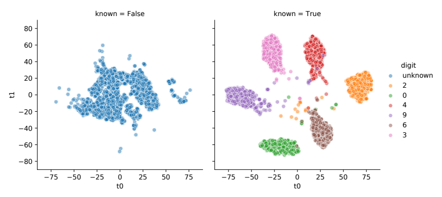
Figure 3c shows the heatmap of MAV values of the simulated open MNIST dataset trained by cross-entropy loss with the MMF extension. We take digits “0”, “2”, “3”, “4”, “6”, “9” as the known classes and the remaining digits as the unknown class. Comparing with the MAV values from the network with standalone cross-entropy loss (Figure 3a), we can find that the MAVs of the known classes become more discriminative from each other, and each of the known classes has its representative feature. (e.g. Z1 for class “0”, Z2 for class “2”). Whereas the MMF extension has less effect on the unknown class, its MAV values are relatively evenly distributed.

Since we recognize the unknown class based on the outlier score described in section 3.3, we analyze both the test samples’ outlier scores from the known classes and the unknown class from the MNIST experiment. Figure 4 shows the histogram of the distributions of the outlier scores in triplet loss experiments and triplet loss with the MMF extension. Compared with standalone triplet loss, adding an MMF extension increases the outlier scores of the unknown class, which pushes the score distributions further away from those of the known classes and results in fewer overlaps between the known classes the unknown class. It is the reduced overlaps that make the known classes and the unknown classes more separable than before. Figure 5 shows the t-SNE (perplexity: 50) plots of the Z-layer representations of the MNIST dataset from the same experiments. With the MMF extension, the known classes and the unknown class are more separate from each other, and the known classes become more disparate than before.

We also perform an ablation analysis for the MMF loss extension to understand the importance of the MMF extension’s two properties. As shown in Table 1, our baselines include (1) standalone loss functions; (2) loss functions with an extension that maximize the most significant feature as Property A (MaxF); (3) loss functions with an extension that minimizes the least significant feature as Property B (MinF). In general, the MMF extension with both properties outperforms the baselines. This result is consistent with our motivation for the two properties at the beginning of Section 3. Moreover, we find that MaxF and MinF extensions can also achieve better performance than standalone loss functions.

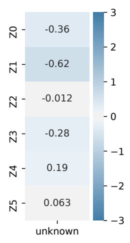


(a) triplet

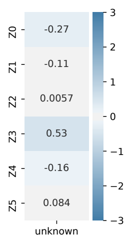


(b) triplet+MMF

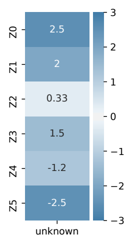
Fig. 5: The t-SNE plots of the MNIST dataset in the experiments of triplet vs. triplet+MMF. The left subplots of (a) and (b) are the representations of the unknown class (a mixture of digits “1”, “5”, “7” and “8”), and the right plots are the representations of the known classes.



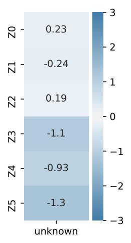
(a) Standalone



(b) +MinF



(c) +MaxF



(d) +MMF

Fig. 6: The heatmap of the unknown class’s MAV in the experiment of cross entropy loss (ce) on the Microsoft Challenge dataset (MC).

While both properties improve AUC scores, Property A (MaxF) has a more significant improvement. Hence, Property A plays a more critical role in AUC improvement than Property B.

To investigate why MinF also helps improve AUC performance, we show the heatmap of the MAV for the unknown class in the experiment of ce on the MC dataset in Figure 6. Comparing Figure 6a and Figure 6b, we observe that MinF reduced the feature magnitudes for the unknown class, thus increased the distance between the known and unknown classes. Similarly, from Figure 6c and Figure 6d, we observe that the feature magnitudes of the unknown class in MMF (MaxF+MinF) are much smaller than the ones in MaxF. The second observation is consistent with the earlier discussion on adding MinF to help MaxF in MMF at the beginning of Section 3. In addition, we observed similar behaviors from other datasets.

5 Conclusion

We introduced a loss function extension for the OSR problem. The extension maximizes the feature with the largest magnitude meanwhile minimizes the one with the smallest magnitude for all the known classes during training so that the learned representations are more discriminative from each other. We have shown that while the known classes are more discriminative from each other, the feature values of unknown classes are less affected by the extension, hence simplifying the open set recognition. We incorporated the proposed extension into classification and representation loss functions and evaluated them in images and malware samples. The results show that the proposed approach has achieved statistically significant improvements for different loss functions.

References

1. Bendale, A., Boult, T.E.: Towards open set deep networks. In: Proc. of the IEEE Conf. on computer vision and pattern recognition. pp. 1563–1572 (2016)
2. Dhamija, A.R., Günther, M., Boult, T.E.: Reducing network agnostophobia. In: Advances in Neural Information Processing Systems 31. pp. 9175–9186 (2018)
3. Gascon, H., Yamaguchi, F., Arp, D., Rieck, K.: Structural detection of android malware using embedded call graphs. In: AISec'13. pp. 45–54 (2013)
4. Hassen, M., Chan, P.K.: Scalable function call graph-based malware classification. In: Proc. ACM Conf. on Data and App. Security and Privacy,. pp. 239–248 (2017)
5. Hassen, M., Chan, P.K.: Learning a neural-network-based representation for open set recognition. Proc. SIAM Intl. Conf. Data Mining pp. 154–162 (2020)
6. Hendrycks, D., Mazeika, M., Dietterich, T.G.: Deep anomaly detection with outlier exposure. In: 7th Intl. Conf. on Learning Representations (2019)
7. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
8. LeCun, Y., Cortes, C., Burges, C.J.: The MNIST database (1999), http://yann.lecun.com/exdb/mnist/
9. Lee, K., Lee, H., Lee, K., Shin, J.: Training confidence-calibrated classifiers for detecting out-of-distribution samples. In: Intl. Conf. Learning Representations (2018)
10. Li, X., Liu, B.: Learning from positive and unlabeled examples with different data distributions. In: European Conf. on Machine Learning. pp. 218–229 (2005)
11. Neal, L., Olson, M., Fern, X., Wong, W.K., Li, F.: Open set learning with counterfactual images. In: European Conf. on Computer Vision. pp. 613–628 (2018)
12. Ortiz, E.G., Becker, B.C.: Face recognition for web-scale datasets. Comput. Vis. Image Underst. **118**, 153–170 (2014)
13. Pidhorskyi, S., Almohsen, R., Doretto, G.: Generative probabilistic novelty detection with adversarial autoencoders. In: NeurIPS 31. pp. 6823–6834 (2018)
14. Ronen, R., Radu, M., Feuerstein, C., Yom-Tov, E., Ahmadi, M.: Microsoft malware classification challenge. CoRR **abs/1802.10135** (2018)
15. Schlegl, T., Seeböck, P., Waldstein, S.M., Schmidt-Erfurth, U., Langs, G.: Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: Information Processing in Medical Imaging. pp. 146–157 (2017)
16. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Conf. on CVPR. pp. 815–823 (2015)
17. Schultheiss, A., Käding, C., Freytag, A., Denzler, J.: Finding the unknown: Novelty detection with extreme value signatures of deep neural activations. In: Pattern Recognition - 39th German Conf. pp. 226–238 (2017)
18. Zhou, Y., Jiang, X.: Android malware genome project (2015), http://www.malgenomeproject.org/