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

Jingyun Jia
Florida Institute of Technology
Melbourne, FL 32901
jiaj2018@my.fit.edu

Philip K. Chan
Florida Institute of Technology
Melbourne, FL 32901
pkc@cs.fit.edu

Abstract

Open set recognition (OSR) is the problem of classifying the known classes, meanwhile identifying the unknown classes when the collected samples cannot exhaust all the classes. There are many applications for the OSR problem. For instance, the frequently emerged new malware classes require a system that can classify the known classes and identify the unknown malware classes. In this paper, we propose an add-on extension for loss functions in neural networks to address the OSR problem. Our loss extension leverages the neural network to find polar representations for the known classes so that the representations of the known and the unknown classes become more effectively separable. 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 terms of training time and model accuracy.

1 Introduction

As classification techniques have achieved great success in various fields in both research and industry, most traditional classification problems, focus on the known classes. However, it is difficult to collect samples that exhaust all classes in the real world. This problem is referred as Open Set Recognition (OSR) [1] or Open Category Learning [3]. That is, OSR attempts to handle "unknown unknowns" for more robust AI systems [3]. For a multinomial classification problem, the OSR problem's objective is to classify the multiple known classes 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 [13], malware classification [7] and medical diagnoses [18].

In this paper, we introduce an MMF loss extension to help the existing loss functions better handle the open set scenario. The MMF extension is inspired by Extreme Value Signatures (EVS) in [20]. Borrowing from a pre-trained neural network for regular classification, EVS uses only the top K activations 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, we utilize a similar idea to directly learn more discriminative features for the known and unknown classes. We name the approach Min Max Feature (MMF) loss extension because we emphasize the features with the smallest and largest magnitudes during network training. 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 problem. Second, we show that MMF achieves statistically significant improvement in AUC when applied to two types of loss functions (classification and representation loss functions) on three datasets from two different domains (images and malware). Third, our results indicate that the combination of MMF and the ii loss function [7] 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. Section 4 shows that the MMF extension can improve different types of loss functions significantly.

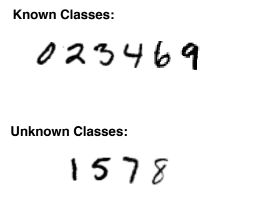
2 Related Work

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. To better discriminate known class and unknown class, unlabeled data is introduced during training in [17, 21]. Dhamija et al. [2] utilize the differences of feature magnitudes between known and borrowed unknown samples as part of the objective function. Shu et al. [22] indicate several manually annotations for unknown classes are required in their workflow. Hendrycks et al. [8] 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.

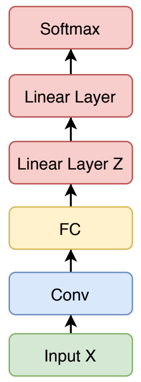
The research works that generate additional data in training data fall in the second category of open set recognition techniques. Most data generation methods are based on GANs. Ge et al. [5] introduce a conditional GAN to generate some unknown samples followed by OpenMax open set classifier. Yu et al. [24] also use the min-max strategy from GANs, generating data around the decision boundary between known and unknown samples as unknown. Neal et al. [12] add another encoder network to traditional GANs to map from images to a latent space. Jo et al. [9] generate unknown samples by a marginal denoising autoencoder that provided a target distribution that is away from the distribution of known samples. Lee et al. [11] 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. [14] propose manifold learning based on training an Adversarial Autoencoder (AAE) to capture the underlying structure of the distributions of known classes. Hassen and Chan [7] 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. Wang et al. [23] design a distance-based loss function with a user-specified margin between classes named pairwise loss to separate different classes. In EVS, Schultheiss et al. [20] investigate class-specific representations for novelty detection tasks. The research work shows the mean representation for each class 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 a challenge. Also, EVS assumes that all the activation values are positive and only looks at the larger ones. We address both limitations in our proposed approach.

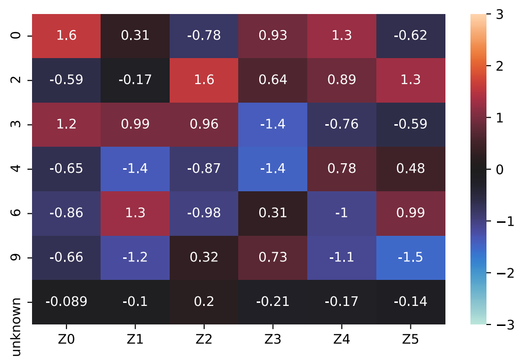
While our proposed approach can be incorporated into different loss functions, we focus on two types of loss functions in this paper: the classification loss functions and the representation loss functions. The objective of classification loss is to lower the classification error of the training data explicitly in the decision layers. One of the widely used classification loss functions is the cross-entropy loss built upon entropy and measure of the relative entropy between two probability distributions, i.e., the actual labels and the predicted labels. The objective of representation loss functions is to learn better representations of training data. The representation loss functions are normally applied to the representation layers, such as triplet loss in [19] and ii loss in [7]. Triplet loss was first introduced for face recognition tasks. It 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. As mentioned above, ii loss was proposed for open set recognition. The objective of ii loss is to maximize the distance between different classes (inter-class separation) and minimize the distance of an instance from its class mean (intra-class spread). So that in the learned representation, instances from the same class are close to each other while those from different classes are further apart.



(a) An open set simulation of the MNIST dataset



(b) A network architecture using cross entropy



(c) The heatmap of the MAV values from the Z-layer of (b)

Figure 1: An example of the MAVs in an open set classification problem with the MNIST dataset.

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.

In a typical neural network, we can consider the hidden layers as different levels of representations of the input. We call the values of the penultimate layer activation vector (AV), and each activation is a learned feature. The mean activation vectors (MAV) is the average of the activation vectors of all the examples for each class. If we measure the feature values in MAVs by their magnitude, the feature values of larger magnitudes are more significant than those with smaller magnitudes. We want the features with larger magnitudes to become even more significant since they can help identify the unknown class. Thus we can either explicitly increase their magnitudes or further decrease the magnitudes of less significant features. 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 from each other while the unknown classes should be less affected.

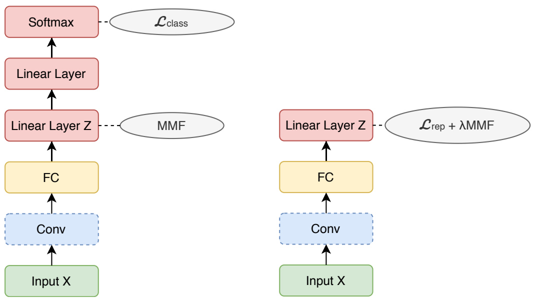
3.1 Learning objectives

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:

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

In the example of Figure 1c, the "max_feature" would be (1.6, 1.6, 1.4, 1.4, 1.3, 1.5). 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:

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



(a) MMF with classification loss function

(b) MMF with representation loss function

Figure 2: An overview of the network architectures of MMF with different types of loss functions. The convolutional layers are optional.

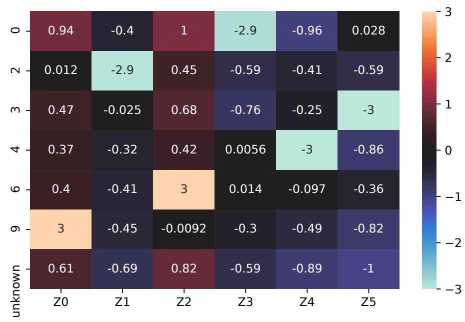


Figure 3: The heatmap of MAV values of the MNIST dataset from the ce+MMF model.

The “min_feature” in the example of Figure 1c would be (0.31, 0.17, 0.59, 0.48, 0.31, 0.32). Then, we maximize the smallest value in “max_feature” directly. In this way, all the values in the “max_feature” would be maximized, 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”, therefore the least significant features for all the known classes would be minimized as Property B. As a result, the proposed MMF extension satisfy both properties:

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

In the example of Figure 1c, we would like to maximize the “1.3” in the “max_feature” and minimize the “0.59” 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 ($max\_feature_{1i} = \max_{1\le j\le F}(\mu_{ij})$) and the lowest values ($max\_feature_{2i} = \max_{1\le j\le F}(-\mu_{ij})$) to form two “max_feature” vectors and later to be maximized at the same time. However, the results (in supplementary materials) 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 proposed extension function and in our experiments.

3.2 Training MMF extension with loss functions

In addition to Properties A and B, the MMF extension can be incorporated into different loss functions for other properties. We focus on two types of loss functions: a) loss functions designed for decision levels such as cross-entropy loss, which is applied to the outputs of the Softmax layer; b) loss functions designed for representation levels such as triplet loss and ii loss, which are applied to the activation vectors. Notably, we combine the MMF extension with these two types of loss functions differently, as Figure 2.

We use the network architecture in Figure 2a to train the network with classification loss functions and the MMF extension simultaneously. During each training iteration, we first extract AVs and generate the representation matrix as shown from line 2 and line 6 in Algorithm 1, then we construct the MMF extension function from the “max_feature” vector and “min_feature” vector as shown from line 7 to line 9. The weights of each layer of the network are first updated to minimize the MMF extension then updated to minimize classification loss functions ($\mathcal{L}_{class}$) using stochastic gradient descent, as shown from line 12 to line 17 in Algorithm 1.

Algorithm 1: Training to minimize MMF with different loss functions

```
Input: $(\boldsymbol{X}, Y)$: Training data and labels
Output: $\mathbb{U}$: Representation matrix for all the known classes; $\boldsymbol{W}$: model parameters
for number of iterations do
    $\boldsymbol{z} \leftarrow$ Extract AVs from the penultimate network layer;
    for $i$ = 1...$C$ do
        $\mu_i \leftarrow mean_j(\boldsymbol{z})$;                          ▷ Compute MAVs for each class
    end
    $\mathbb{U} \leftarrow STACK(\mu_i)$;
    $max\_feature \leftarrow get\_max\_feature(\mathbb{U})$;              ▷ As in Equation 1
    $min\_feature \leftarrow get\_min\_feature(\mathbb{U})$;              ▷ As in Equation 2
    $MMF \leftarrow \max(min\_feature) - \min(max\_feature)$;            ▷ As in Equation 3
    if representation loss function $\mathcal{L}_{rep}$ then
        $\mathcal{L} \leftarrow \mathcal{L}_{rep} + \lambda MMF$ ;
    else
        $\mathcal{L} \leftarrow MMF$;
    end
    $\boldsymbol{W} \leftarrow SGD(\boldsymbol{W}, \mathcal{L})$;
    if classification loss function $\mathcal{L}_{class}$ then
        $\boldsymbol{W} \leftarrow SGD(\boldsymbol{W}, \mathcal{L}_{class})$;
end
return $\mathbb{U}$, $\boldsymbol{W}$;
```

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:

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

where $\mathcal{L}_{rep}$ is a representation loss function, and λ is a parameter that strikes a balance between the representation loss function and the MMF extension. Figure 2b 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, as shown from line 10-11 and line 15 in Algorithm 1.

3.3 Open Set Recognition (OSR)

A typical open set recognition task solves two problems: one is classifying the known classes, the other is identifying the unknown class. From the representation level, the instances from the same class are close to each other, while those from different classes are further apart. Thus the instances from the unknown class should be far away from any known classes. Under this property, we propose the outlier score in test phase:

$$outlier\_score(x) = \min_{1 \le i \le C} \|\mu_i - z\|_2^2, \tag{5}$$

where z is the learned representation of test sample x, μ_i is the MAV of the known class i. There are multiple ways to set the outlier threshold. Here, we sort the outlier score of the training date in ascending order and pick the 99 percentile outlier score value as the outlier threshold. Then, for the C known classes, we predict the class probability $P(y = i|x)$ for each class. When a network is trained on classification loss function, such as cross entropy, the $P(y = i|x)$ is the output of the Softmax layer as Figure 2a. Whereas in the case of a network without softmax layer such as Figure 2b, we calculate $P(y = i|x)$ as:

$$P(y = i|x) = \frac{e^{-\|\mu_i - z\|_2^2}}{\sum_{j=1}^{C} e^{-\|\mu_j - z\|_2^2}} \tag{6}$$

In summary, a test instance is recognized as "unknown" if its outlier score is greater than the threshold, otherwise it is classified as the known class with the highest class probability:

$$y = \begin{cases} unknown, & \text{if } outlier\_score(x) > threshold \\ \underset{1 \leq i \leq C}{\operatorname{argmax}} P(y = i|x), & \text{otherwise} \end{cases} \tag{7}$$

4 Experimental Evaluation

We evaluate the MMF extension with simulated open-set datasets from the following three datasets.

MNIST [15] 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 classes of digits as the known classes participant in the training, while the rest are treated as the unknown class only existing in the test set.

Microsoft Challenge (MC) [10] 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 [6] for the experiment. Again, to simulate an open-world dataset, we randomly pick six classes of digits as the known classes participant in the training, while the rest are considered as unknowns.

Android Genome (AG) [25] consists of malicious android apps from many families in different sizes. We use nine families (986 samples) with the relatively larger size for the experiment so that they could be fairly split into the training set, the test set, and the validation set. we first use [4] to extract the function instructions and then extract the FCG features as in [6]. Same as the MNIST and the MC dataset, we randomly pick six classes of digits 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 Evaluation Network Architectures and Evaluation Criteria

We evaluate the MMF extension associated with two types of loss functions mentioned above: 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 [7] and triplet loss [19] as the examples of representation loss functions, resulting in three pairs of comparisons: regular cross-entropy loss (ce) vs. cross-entropy loss with the MMF extension (ce+MMF); regular ii loss (ii) vs. ii loss with the MMF extension (ii+MMF); regular triplet loss (triplet) vs. triplet loss with the MMF extension (triplet+MMF). We implement the network architectures in Figure 2 using Tensorflow.

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-polling layers with kernel size (3, 3) and strides (2, 2) after each convolutional layers. 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 of six dimensions in our experiment. We use the Relu activation function for all the non-linear layers and set the keep probability of Dropout as 0.2 for the fully connected layers. We use Adam optimizer with a learning rate of 0.001. The experiment for the MS Challenge dataset follows similar network architecture as the MNIST dataset. The differences are: the padded input layer is of size (67, 67). We only use one fully connected layer instead of two after the convolutional component. 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 keep probability of 0.9 for the fully connected layers and 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 3.3, we use contamination ratio of 0.01 for the threshold selection.

For each dataset, we simulate three different groups of open sets 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. 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 achieve statistically significant improvements on different loss functions.

Table 1: The average AUC scores of 30 runs at 100% and 10% FPR of three quadruples: regular loss functions (Regular), with the MMF extension (+MMF), only maximizing the "max_feature" (+MaxF) and only minimizing the "min_feature" (+MinF). The underlined values are statistical significant better than the regular loss functions via t-test with 95% confidence. The values in bold are the highest value in each quadruple.

		ce				ii				triplet			
	FPR	Regular	+MMF	+MaxF	+MinF	Regular	+MMF	+MaxF	+MinF	Regular	+MMF	+MaxF	+MinF
MNIST	100%	0.9255	$\underline{0.9479}$	$\underline{\mathbf{0.9515}}$	0.9393	0.9578	$\underline{\mathbf{0.9649}}$	0.9579	0.9607	0.9496	$\underline{\mathbf{0.9585}}$	0.9480	0.9404
	10%	**0.0765**	0.0744	0.0761	0.0751	0.0821	$\underline{\mathbf{0.0842}}$	0.0826	0.0830	0.0750	$\underline{\mathbf{0.0796}}$	0.0777	0.0739
MC	100%	0.9148	$\underline{\mathbf{0.9500}}$	$\underline{0.9387}$	$\underline{0.9352}$	0.9385	$\underline{\mathbf{0.9461}}$	0.9407	0.9397	0.9240	$\underline{\mathbf{0.9430}}$	0.9317	0.9178
	10%	0.0530	$\underline{\mathbf{0.0635}}$	$\underline{0.0600}$	$\underline{0.0588}$	0.0627	$\underline{\mathbf{0.0656}}$	0.0629	0.0619	0.0565	$\underline{\mathbf{0.0622}}$	0.0563	0.0546
AG	100%	0.7506	$\underline{\mathbf{0.8205}}$	$\underline{0.8152}$	0.7501	0.8427	0.8694	0.8763	**0.8831**	0.8271	**0.8379**	0.8203	0.8256
	10%	0.0058	$\underline{0.0148}$	$\underline{\mathbf{0.0163}}$	$\underline{0.0036}$	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 three pairs: regular loss functions (Regular) and with the MMF extension (+MMF) for the "Known", "Unknown" and all classes. The underlined values show statistically significant improvements (t-test with 95% confidence) comparing with the regular loss functions. The values in bold are the highest value in each column.

		MNIST			MC			AG		
		Known	Unknown	Overall	Known	Unknown	Overall	Known	Unknown	Overall
ce	Regular	0.7596	0.7561	0.7591	0.8881	0.6643	0.8562	0.5346	0.0033	0.4587
	+MMF	$\underline{0.8504}$	$\underline{\mathbf{0.7902}}$	$\underline{0.8809}$	$\underline{0.9090}$	**0.7963**	$\underline{0.8929}$	0.5555	$\underline{0.1142}$	0.4925
ii	Regular	0.9320	0.8833	0.9250	0.9128	0.7257	0.886	0.6349	0.6677	0.6396
	+MMF	**0.9373**	0.8916	**0.9308**	$\underline{\mathbf{0.9210}}$	$\underline{0.7680}$	$\underline{\mathbf{0.8991}}$	**0.6407**	**0.7251**	**0.6528**
triplet	Regular	0.9103	0.8302	0.8989	0.8998	0.7018	0.8715	0.5929	0.6323	0.5986
	+MMF	$\underline{0.9239}$	$\underline{0.8625}$	$\underline{0.9152}$	0.9064	0.7213	0.8800	0.6005	0.6895	0.6132

4.2 Experimental Results

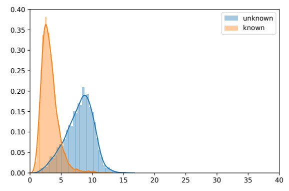
We compare the model performances of cross-entropy loss, ii loss, and triplet loss before and after applying the MMF extension. Table 1 shows the AUC scores of the models in the three datasets; mainly, we focus on comparing the "Regular" with the corresponding "+MMF" subcolumns. We observe that with the MMF extension, the AUC scores of the loss functions have achieved statistically significant improvements in 13 out of 18 cases (3 loss functions$\times$3 datasets$\times$2 FPR values).

Table 2 shows the average F1 scores for the three 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 the loss functions with the MMF extension have better results than their corresponding regular 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 7 out of 9 F1 scores. Particularly, it achieves the highest Overall F1 scores for all three datasets.

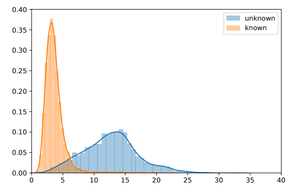
Table 3 shows the comparison of the average training time of the 30 runs for the MNIST dataset with 5000 iterations via NVIDIA Tesla K80 GPU on AWS. As we can see, adding the MMF extension almost doubles the training time of using regular cross-entropy. While for ii loss and triplet loss, adding the 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 cross-entropy loss, and an extra backpropagation step, both of which take more time. We also observe that with our MMF extension, ii loss has the fastest training time among three loss functions. Overall F1 scores and training time indicate that "ii+MMF" is the most accurate and efficient combination.

4.3 Analysis

Figure 3 shows the heatmap of MAV values of the simulated open MNIST dataset trained by cross-entropy loss with the MMF extension. Same as Figure 1a , we take digits "0", "2", "3", "4", "6", "9" as the known classes and the rest four digits as the unknown class. Comparing with the MAV values from the network with regular cross-entropy loss (Figure 1c), 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 most significant feature. (e.g. Z3 for class "0", Z1 for class "2"). Whereas the MMF extension has less effect on the unknown class, and its MAV values are relatively evenly distributed.



(a) triplet

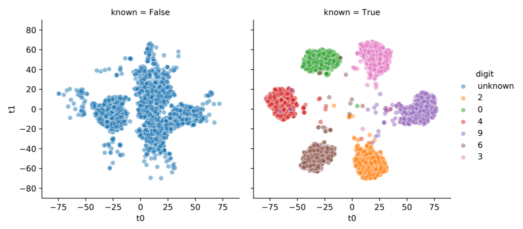


(b) triplet+MMF

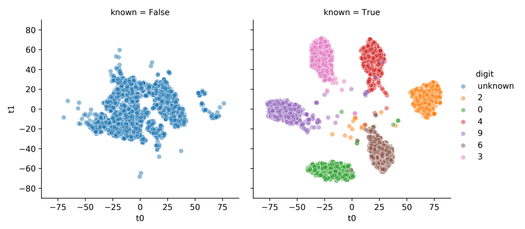
Figure 4: The distributions of outlier scores for the known and unknown classes of the MNIST dataset in the experiments of triplet vs. triplet+MMF.

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



(a) triplet



(b) triplet+MMF

Figure 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.

Since we recognize the unknown class based on the outlier score described in section 3.3, we analyze the outlier scores of both the test samples 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 the experiments of triplet loss and triplet loss with the MMF extension. Comparing with using regular 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 and 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. We can see that with the MMF extension, not only the known classes and the unknown class are more separate from each other, 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) regular 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. Meanwhile, the most significant feature in Property A plays a more critical role than the least significant feature in Property B.

5 Conclusion

We introduced an add-on 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 presentations 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 both 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.

Broader Impact

The motivation of our work is to identify the unknown malware classes better. The traditional malware classification systems require engineers to identify and update malware signatures manually. In contrast, the OSR system can automatically identify the known classes as well as the unknown classes. Besides malware classification, our work can also be applied to other areas such as face recognition and medical diagnoses. However, it could also be misused. For example, in government surveillance systems, known classes could be "desirable groups of people", hence "undesirable groups of people" could be identified and possibly discriminated. Moreover, the fact that deep learning classification systems have become popular and widely applied comes with the risk of adversarial attacks. For example, our work can detect unknown malware and classify known malware. However, a slight perturbation in adversarial samples would make the system misclassify malware samples. Many deep learning applications are suffering from this problem, and it endangers today's digital world. Recent work in open set recognition has shown promising results against adversarial attacks [16], they show that the designed OSR system is more robust to attacks. More work is needed to show that our work is robust to adversarial attacks and potentially be applied to adversarial attack detection systems. In addition to the misuses and adversarial attacks, another concern is that just like most automatic classification systems, our work can reduce the essential cost of human labor and may have a potential impact on job loss. Nevertheless, there are limitations for both machines and humans, and finding a balance between them is vital for real-world applications as well as further research.

Acknowledgements and Funding Disclosures

This work is partially supported by grants from Amazon and Rockwell Collins to Dr. Philip Chan. We also thank Dr. Mehadi Hassen for sharing his source code and comments on this paper.

References

[1] BENDALE, A., AND BOULT, T. E. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 1563–1572.

[2] DHAMIJA, A. R., GÜNTHER, M., AND BOULT, T. E. Reducing network agnostophobia. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada* (2018), pp. 9175–9186.

[3] DIETTERICH, T. G. Steps toward robust artificial intelligence. *AI Magazine 38*, 3 (2017), 3–24.

[4] GASCON, H., YAMAGUCHI, F., ARP, D., AND RIECK, K. Structural detection of android malware using embedded call graphs. In *AISec'13, Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security* (2013), pp. 45–54.

[5] GE, Z., DEMYANOV, S., AND GARNAVI, R. Generative openmax for multi-class open set classification. In *British Machine Vision Conference* (2017).

[6] HASSEN, M., AND CHAN, P. K. Scalable function call graph-based malware classification. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy,* (2017), pp. 239–248.

[7] HASSEN, M., AND CHAN, P. K. Learning a neural-network-based representation for open set recognition. *Proc. SIAM Intl. Conf. Data Mining* (2020), 154–162.

[8] HENDRYCKS, D., MAZEIKA, M., AND DIETTERICH, T. G. Deep anomaly detection with outlier exposure. In *7th International Conference on Learning Representations* (2019).

[9] JO, I., KIM, J., KANG, H., KIM, Y.-D., AND CHOI, S. Open set recognition by regularising classifier with fake data generated by generative adversarial networks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), IEEE, pp. 2686–2690.

[10] LECUN, Y., CORTES, C., AND BURGES, C. J. The MNIST database, 1999.

[11] LEE, K., LEE, H., LEE, K., AND SHIN, J. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *6th International Conference on Learning Representations* (2018).

[12] NEAL, L., OLSON, M., FERN, X., WONG, W.-K., AND LI, F. Open set learning with counterfactual images. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 613–628.

[13] ORTIZ, E. G., AND BECKER, B. C. Face recognition for web-scale datasets. *Comput. Vis. Image Underst. 118* (2014), 153–170.

[14] PIDHORSKYI, S., ALMOHSEN, R., AND DORETTO, G. Generative probabilistic novelty detection with adversarial autoencoders. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada* (2018), pp. 6823–6834.

[15] RONEN, R., RADU, M., FEUERSTEIN, C., YOM-TOV, E., AND AHMADI, M. Microsoft malware classification challenge. *CoRR abs/1802.10135* (2018).

[16] ROZSA, A., GÜNTHER, M., AND BOULT, T. E. Adversarial robustness: Softmax versus openmax. In *British Machine Vision Conference 2017, BMVC 2017, London, UK, September 4-7, 2017* (2017).

[17] SAITO, K., YAMAMOTO, S., USHIKU, Y., AND HARADA, T. Open set domain adaptation by backpropagation. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 153–168.

[18] SCHLEGL, T., SEEBÖCK, P., WALDSTEIN, S. M., SCHMIDT-ERFURTH, U., AND LANGS, G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *Information Processing in Medical Imaging - 25th International Conference* (2017), pp. 146–157.

[19] SCHROFF, F., KALENICHENKO, D., AND PHILBIN, J. Facenet: A unified embedding for face recognition and clustering. In *Conference on Computer Vision and Pattern Recognition* (2015), IEEE, pp. 815–823.

[20] SCHULTHEISS, A., KÄDING, C., FREYTAG, A., AND DENZLER, J. Finding the unknown: Novelty detection with extreme value signatures of deep neural activations. In *Pattern Recognition - 39th German Conference* (2017), pp. 226–238.

[21] SHU, L., XU, H., AND LIU, B. Unseen class discovery in open-world classification. *arXiv preprint arXiv:1801.05609* (2018).

[22] SHU, Y., SHI, Y., WANG, Y., ZOU, Y., YUAN, Q., AND TIAN, Y. ODN: Opening the deep network for open-set action recognition. In *2018 IEEE International Conference on Multimedia and Expo (ICME)* (2018), IEEE, pp. 1–6.

[23] WANG, Z., KONG, Z., TAO, H., CHANDRA, S., AND KHAN, L. Co-representation learning for classification and novel class detection via deep networks. *arXiv preprint arXiv:1811.05141* (2018).

[24] YU, Y., QU, W., LI, N., AND GUO, Z. Open category classification by adversarial sample generation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence* (2017), pp. 3357–3363.

[25] ZHOU, Y., AND JIANG, X. Android malware genome project, 2015.