Unsupervised Open Set Recognition using Adversarial Autoencoders

***Abstract*—We propose an unsupervised open set recognition approach that uses an Adversarial Autoencoder (AAE) for clustering and combines it with ii-loss to learn a representation that facilitates open set recognition. We evaluate our approach on two malware datasets. To show the applicability of the proposed method outside the malware domain we also evaluate it an image dataset.**

I. Introduction

Scheirer et al. [1] define an open set recognition scenario as one where an "incomplete knowledge of the world is present at training time, and unknown classes can be submitted to an algorithm during testing". In other words, the classes seen during testing may not have been seen in training; unlike closed set recognition where the assumption is that the same set of classes are seen both during training and test. Many research work on open set recognition [2]–[4] propose solutions for a supervised open set recognition task where the training data is labeled. However, there are scenarios where open set recognition is needed in an unsupervised setting. That is, in the unsupervised setting, the training data are not labeled with classes, and the test data could belong to classes beyond those "latent" classes in the training data.

Consider a scenario where the training set is comprised of instances from multiple data generating distributions (we will refer to these as known or training data distributions). Furthermore, test data can contain instances from unknown data generating distribution never seen in the training set. Unsupervised open set recognition is the task of identifying whether an instance is from unknown data distributions or the known data distributions in the absence of labeled training data. Wang et al. [5] highlight one such scenario for the task of person re-identification from surveillance camera footages in public space. Another case where unsupervised open set arises is in malware clustering [6]–[9]. Rieck et al. [8] identified the necessity of having open set recognition capability and proposed a malware clustering approach and an associated outlier score for recognizing instances that did not belong to the already identified clusters.

Our contributions include: (1) we propose an unsupervised open set recognition approach that uses an Adversarial Autoencoder (AAE) [10] for clustering and combine it with ii-loss, proposed in [11], to learn a representation that facilitates open set recognition. (2) Our proposed approach gives a statistically significant improvement over other approaches on three evaluation datasets; two malware datasets (as the primary focus of this work is the security domain) and one image dataset. (3) We show anomaly or outlier detection algorithms, such as Isolation Forest, do not perform well on unsupervised open set scenarios. (4) We show the applicability of our approach to image recognition domain by evaluating on handwritten digit dataset (MNIST).

II. Related Works

Open set recognition is starting to get considerable attention in recent literature. Dietterich [12] even presents it as one of the challenges that need to be addressed to build a robust AI system. Most of the research in this area [1]–[4], [13] so far has focused on supervised open set recognition. Scheirer et al. [1] formalize open space risk and propose a one-class open set recognition (they call a 1-vs-set machine) as an SVM where the positive region is bounded by two parallel planes. Scheirer et al. [4] further develop this idea to address a multi-class open set recognition and provide a probabilistic decision score for rejecting instances as unknown. Bendale and Boult [3] present a deep neural network based multi-class open set recognition approach which redistributes the activation vector of the last neural network layer to compute the pseudo-activation for the unknown class and provide the probability of the unknown class. On the unsupervised open set recognition side, Want et al. [5] present an approach for learning to identify if individuals in surveillance footages that are of the same individuals in a target list without the need for a labeled data.

Another related area is anomaly or outlier detection. Proximity-based anomaly detection approaches such as [14], [15], for example, use the distance of an instance to its k-nearest neighbor as its anomaly/outlier score. Others such as the widely used LOF algorithm [16] use the relative density of the neighborhood around an instance as a measure of the instances outlier score. Isolation Forest [17], a random forest like anomaly detection algorithm, can also be thought of as a density-based approach. Although isolation forest does not directly calculate the density of an instance, it measures how anomalous an instance is by how difficult it is to isolate that instance from other instances around it through a random recursive partitioning. The denser the area around an instance the more difficult it is to separate it hence the less likely for it to be an anomaly. Clustering-based anomaly detection has also been proposed [18]. Proximity-based, density-based, and clustering-based anomaly detection approaches rely on the use of the appropriate distance function which can be difficult to find for the original feature space. As a result, typical anomaly detection methods do not perform well on open set recognition as we show in our evaluations.

III. Background

a) Adversarial Autoencoder (AAE): Makhzani et al. [10] propose an Adversarial Autoencoder which turns the decoder network of an autoencoder into a generative model. In their proposed approach an adversarial autoencoder, shown in Part A of Figure 1, is trained on two objectives (1) the traditional reconstruction error criteria and (2) adversarial training criteria. The reconstruction objective, Eq. 1, aims to train the encoder network to encode the original input into latent variables and the decoder network to reconstruct the original input from the latent variables. The task of the adversarial criteria, Eq. 2 and 3, is to match the latent variable posterior distribution to an arbitrary prior. The discriminator network D, minimizes the classification error, in the form of a standard cross-entropy function, when assigning the correct labels to the real samples from the Gaussian distribution $p(z\prime)$ and the fake samples from the generator G. The generator G on the other hand, tries to fool the discriminator into labeling the fake samples as real.

$$reconstruction\_loss = \frac{1}{2N}\sum_{i=1}^{N}\left\|\hat{\vec{x}_i} - \vec{x_i}\right\|_2^2 \tag{1}$$

where N is number of instances and $\hat{\vec{x}_i}$ is the reconstruction of the training sample $\vec{x_i}$.

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{z\prime\sim p(\vec{z}\prime)}[\log D(\vec{z}\prime)] - \frac{1}{2}\mathbb{E}_{\vec{x}_i\sim p_{data}()}[\log(1 - D(G(\vec{x}_i))] \tag{2}$$

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{x_i\sim p_{data}()}[\log D(G(x_i))] \tag{3}$$

where $\vec{z}\prime$ is a sample drawn from an arbitrary distribution $p(\vec{z}\prime)$, $\vec{x}_i$ is training sample drawn from the data distribution $p_{data}()$, and finally D and G are the Discriminator and Generator networks.

In case of the AAE shown in Part A of Figure 1, the encoder network, and the z-layer together form the generator G. The discriminator D, on the other hand, is a separate network which we refer to as "Gaussian Discriminator" in the Figure 1. We use the term "Gaussian Discriminator" to signify that the real samples are drawn from a Gaussian distribution (i.e. $p(\vec{z}\prime)$ in Equation 2 is now $\mathcal{N}(0, I)$). On the other hand, the fake samples are from the output of the z-layer. During each training iteration the training alternates between two phases. In the first phase (called the reconstruction phase), the encoder network, decoder network, and the z-layer weights are updated to minimize the reconstruction error. Then in the second phase (the adversarial phase), the discriminator network weights and the generator network weights are updated by minimizing Eq. 2 and 3, respectively.

b) Clustering Variant of Adversarial Autoencoder (AAE): Makhzani et al. also demonstrate a clustering variant of the adversarial autoencoder where the encoder network is connected to the z-layer and the y-layer. The y-layer, which is a softmax layer, is a probability distribution over the cluster labels,shown in Part B of Figure 1. There is an additional adversarial network (which we will refer to as "Categorical Discriminator") for regularizing the y-layer such that the cluster prediction match a Categorical distribution (i.e., one-hot encoding). Each training iteration of the clustering variant of AAE consists of three phases. The first two phases are the same as the two phases for AAE. The third is the categorical adversarial phase. In the context of this phase, the Categorical Discriminator represents the D network; whereas the encoder and y-layer constitute the generator G network. In this phase, the weights of the D and the G networks are updated to minimize Eq. 2 and 3, respectively. However, in this case the real samples are drawn from a Categorical distribution (i.e., $p(\vec{z}\prime)$ in Eq. 2 refers to a Categorical distribution); where as the fake samples come from the output of the y-layer.

c) II-Loss: The aim of ii-loss, proposed in [11], is to learn a neural-network-based representation that facilitates open set recognition by fulling two properties: (P1) instances from the same class are close to each other; (P2) instances from different classes are further apart. The intuition is that as the known classes are further apart from each other, unknown class instances will have more space to occupy between them making the task of recognizing unknown class instances easier. Given training data consisting of set of instance X and labels Y, and a neural network g, the learned representation of instance $\vec{x_i}$ is $\vec{z_i} = g(\vec{x_i})$. The neural network is trained to minimize ii-loss which defined as:

$$\textit{ii-loss} = \underbrace{\Big(\frac{1}{N}\sum_{j=1}^{K}\sum_{i=1}^{|C_j|}\|\vec{\mu_j} - \vec{z_i}\|_2^2\Big)}_{\text{intra\_spread}} - \underbrace{\Big(\min_{1\le m\le K}\ \min_{m+1\le n\le K}\|\vec{\mu_m} - \vec{\mu_n}\|_2^2\Big)}_{\text{inter\_sparation}} \tag{4}$$

where $|C_j|$ is the number of training instances in class C_j, N is the number of training instances, K is the number of known classes, and $\vec{\mu_j} = \frac{1}{|C_j|}\sum_{i=1}^{|C_j|}\vec{z_i}$ is the mean of class C_j. To satisfy P1, we minimize the distance the projection $\vec{z}$ has from the instance's class center, $intra\_spread$. To satisfy P2, we maximize the distance between the closest pair of class centers, $inter\_sparation$.

IV. Unsupervised Open Set Recognition

The original $\vec{z}$ in AAE is used to learn a representation for reconstruction, and it is regularized to follow a normal distribution by the adversarial network so that it can be used for the generative model. As a result, we see that in Figure 2a all instances from the different clusters are projected together into the same region. Additionally, instances from the unknown data distributions are projected in the same region, as shown in figure 2c..

We propose combining ii-loss with AAE to learn a continuous representation for facilitating open set recognition. Since the adversarial regularization works in contrary to ii-loss and as a result, degrades open set recognition performance; we propose to learn a different continuous representation $\vec{z\_ii}$. In $\vec{z\_ii}$, instances from the same cluster are closer to each other while instances from different clusters are further apart, as shown in Figure 2b. We will refer to this new representation

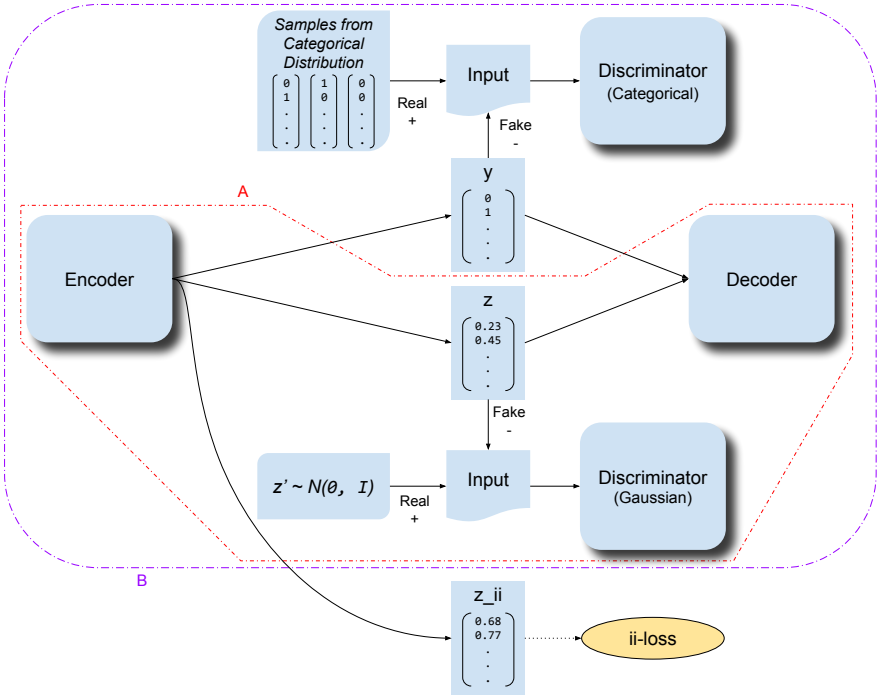


Fig. 1: AAE-II:Adversarial autoencoder architecture for unsupervised open set recognition.

$\vec{z\_ii}$, shown in Figure 1, as a z_ii-layer. The z_ii-layer shares the same encoder network as the z-layer and the y-layer. However, it is not given as input to the decoder as shown in Figure 1.

The original ii-loss in [11] was used for supervised open set recognition, where the training class labels are used for calculating ii-loss. In this case, however, the open set recognition task at hand is unsupervised. Hence, the objective of ii-loss needs to be changed accordingly. We propose to use ii-loss to project similar instances from the same cluster closer together while projecting different clusters further apart. We can use the same ii-loss formulated in Eq. 4. However, in this case, C_j in Eq. 4 will refer to a cluster label instead of class labels and $\vec{z}_i$ in the equation will refer to $vecz\_ii$. A cluster label (or cluster-id) of an instance is computed as $cluster\_id = \underset{1 \leq j \leq K}{\operatorname{argmax}}\, y[j]$ where K is the number of clusters or the dimension of the y-layer.

The training of the AAE-II is carried out in two stages as shown in Algorithm 1. In the first stage (lines 10-11), we train the AAE network with updates from the reconstruction phase, gaussian adversarial phase and categorical adversarial phase. During this training, we monitor the percent of change in cluster membership among the training instances during each training iteration. We can stop the first stage when the change in cluster membership converges to a value less than 1% or when a pre-set training iteration is reached. In the second stage (lines 13-15), we add the z_ii-layer to the network and start updating the z_ii-layer and the encoder network to minimize ii-loss. We also continue the reconstruction, gaussian adversarial, and categorical adversarial phases as in the first stage. After the training is completed, the cluster means (centers), at the

Algorithm 1: Training AAE-II

Input : X: Training data

1 **Function** `AAE_Update()`**:**
2 Sample a mini-batch X_{batch} from X
3 Update Encoder, z-layer, y-layer, and Decoder to minimize reconstruction loss, Eq. 1.
4 Update Gaussian Discriminator to minimize Eq. 2 where $G(X_{batch}) = L_z(Enc(X_{batch}))$ and $p(\vec{z}\prime) = \mathcal{N}(0, I)$
5 Update Categorical Discriminator to minimize Eq. 2 where $G(X_{batch}) = L_y(Enc(X_{batch}))$ and $p(\vec{z}\prime) = \mathcal{C}at()$
6 Update Encoder and z-layer to minimize Eq. 3 where $G(X_{batch}) = L_z(Enc(X_{batch}))$
7 Update Encoder to y-layer to minimize Eq. 3 where $G(X_{batch}) = L_y(Enc(X_{batch}))$
8 **return**;
9 // Stage One
10 **for** *number of stage one iterations* **do**
11 `AAE_Update()`
12 // Stage Two
13 **for** *number of stage two iterations* **do**
14 `AAE_Update()`
15 Update Encoder and z_ii-layer to minimize ii-loss, Eq. 4
16 Calculate cluster means ($\vec{\mu}_j$) and save as part of the model.

L_z denotes the z-layer, L_y denotes the y-layer, $Enc()$ denotes the encoder network , and $Cat()$ denotes a categorical distribution

z_ii-layer, are calculated for each cluster using all the training instances and stored as part of the model.

a) Outlier Score and Threshold Estimation: To measure the degree to which a test instance $\vec{x}$ is an outlier, we get the z_ii-layer output for this instance ($\vec{z\_ii}$) and then measure how far this vector is from each cluster mean. The distance from the closest cluster center (mean) is then used as an outlier score.

$$outlier\_score(\vec{x}) = \min_{1 \leq j \leq K} \left\| \vec{\mu}_j - \vec{z\_ii} \right\|_2^2 \quad (5)$$

where $\vec{z\_ii}$ is the output of the z_ii layer for input $\vec{x}$, and $\vec{\mu}_j$ is the mean of cluster j.

To determine whether a test instance is an outlier, a threshold value on the outlier score is needed. We estimate this threshold based on the outlier score of the training data. An assumption is made on how much of the training data to consider as outliers using the contamination ratio. For example, if the contamination ratio is set to be 2%, then the 98 percentile largest outlier score of the training data is set as the threshold value. During testing any instance that has an outlier score higher than the threshold is predicted to be an outlier.

V. Evaluation

a) Dataset and Simulating Open Set Dataset: We evaluated our proposed approach on three datasets. The first dataset is the Microsoft Malware Challenge Dataset [19] of labeled disassembled windows malware samples from 9 families. We use 10260 (out of 10868) samples that our disassemble file parser can properly process. The majority class has 2936 samples whereas the minority class has 34 samples with the median class size of 1012 samples. The second dataset is the Android Malware Genome Project dataset [20]. We use 986 samples from 9 families that each has at least 40 samples. The majority class in this dataset has 309 samples whereas the minority class has 46 samples with the median class size being 69. The malware features used in both malware dataset are based on function call graph features proposed in [21]. In both cases, we use minhash to cluster the functions into 64 clusters based on the unigram of the function opcodes. The cluster-ids are then used to label the functions, and the vertex and edge features are extracted. To evaluate the applicability of the proposed approach to domains outside malware classification, we also evaluate it on the MNIST handwritten digit dataset. All three datasets are available online on their respective websites.

We simulate open set datasets from the original evaluation datasets. All three of the original datasets are labeled; however, we do not use the labels during the training of the evaluated approaches. The labels are only used to create the open set datasets and calculating performance metrics. We start by randomly selecting K classes whose instances are going to be present in the training set. In other words, the training set is made up of instances of K data distributions; we will refer to these as known or training data distributions. In case of the MS and Android datasets, we choose 75% of instances from the K classes to be part of the open set training set and add the remaining to the open set test set. All the instances that are not part of the K classes are added to the open set test set. In case of the MNIST dataset, we select all the instances from the K classes in the original MNIST training dataset to be part of the open set training set. All the instances from the original MNIST test set are used in the open set test dataset (i.e., this contains the K training classes and the other classes that are not part of the training set). In all cases the classes labels are not used for training, they are only used for calculating unsupervised open set recognition performance metrics.

b) Evaluated Approaches: To show the performance of outlier detection algorithms on unsupervised open set recognition, we evaluate the approach proposed in contrast to Isolation Forest [17]. We also evaluate how well clustering-based outlier detection performs in an unsupervised open set scenario. Similar to what is proposed by Rieck et al. [8], we use a KMeans-based approach where the distance from cluster centroid is used for detecting unknown data distribution instances.

In case of the Android and MS datasets, both the KMeans-based and Isolation Forest are given the malware FCG features as input features. However, in case of the MNIST dataset, the KMeans-based approach and Isolation Forest understandably do not perform well when given the raw image pixels as input features. So instead we first use an autoencoder to transform the pixels into an intermediate representation and then feed this intermediate representation to the two algorithms.

Our proposed approach was implemented using TensorFlow. The details of our implementation are available in the Supplemental Materials. In case Android and MS dataset the FCG features are given as input features to our approach. Whereas in case of MNIST dataset our approach takes the raw images as input.

A. Detecting Instances from Unknown Data Distributions

For evaluations, three open set datasets are randomly created from each the three datasets as discussed in the earlier Section. We then run 10 experiments for each of the three open set dataset, which means we have a total of 30 experiments for each original dataset. The objective of these experiments is to perform unsupervised open set recognition. In other words, discriminate test instances that belong to the known data distributions (i.e., the data distributions present in the training set) from instances that belong to unknown data distributions (i.e., the data distributions not seen in the training set).

We report the results of these experiments first using Area Under the Curve (AUC), which measures the unsupervised open set recognition performance independent of the outlier threshold. We use a t-test to measure the statistical significance of the difference in performance AUC values. Table I shows the 30-run average AUC. As shown in Table I, our proposed approach (AAE-II) has significantly higher average AUC for MNIST and MS dataset both up to 100% False Positive Rate (FPR) and up to 10% FPR. In case of the Android dataset, the KMeans-based approach has higher average AUC up to 100% FPR over AAE-II although the improvement is not statistically

TABLE I: 30-run average AUC of discriminating instances of unknown data distributions from instances of known data distributions. When calculating the AUC, the positive label represented instances from unknown data distributions. The underlined average AUC values are higher with statistical significance (p-value less than 0.05 with a t-test) compared to the values that are not underlined on the same row. The average AUC values in **bold** are the largest average AUC values in each row.

	FPR	Isolation Forest	KMeans-based	AAE-II
MNIST	100%	0.619(±0.085)	0.841(±0.078)	**0.898**(±0.029)
	10%	0.008(±0.003)	0.031(±0.013)	**0.058**(±0.010)
MS	100%	0.547(±0.167)	0.568(±0.107)	**0.829**(±0.065)
	10%	0.014(±0.017)	0.011(±0.009)	**0.027**(±0.017)
Android	100%	0.716(±0.050)	0.789(±0.016)	**0.817**(±0.139)
	10%	0.011(±0.002)	0.026(±0.009)	**0.029**(±0.016)

significant. When considering AUC up to 10% FPR AAE-II has a higher value. Again the improvement is not statistically significant.

We now present the result of these experiment from a different perspective where a threshold value is estimated. The threshold is then used to predict whether an instance belongs to known data distribution or it does not. We present the results of this prediction in Table II. In all three datasets, our proposed approach records statistically significant improvement in F-Score compared to the other approaches.

VI. Discussion

We investigated the effect ii-loss on clustering performance of AAE by comparing the clustering performance of AAE with and without ii-loss. We report these results in terms of the cluster homogeneity as defined by Rosenberg and Hirschberg [22]. The homogeneity values have a range of 0.0 to 1.0. A homogeneity of 1.0 indicates that a cluster is purely made up of instances from a single class, which is a desirable result. For the homogeneity computation, elements in cluster i are assigned the label of the training instance that maximizes the probability of belonging to that cluster. As shown in Table III, the clustering performance with and without ii-loss remain similar. According to the results from a t-test, the differences were not statistically significant since the p-values from all three rows in the table are higher than 0.24.

In our approach, the use of a separate z_ii-layer was driven by the realization that the original z-layer of AAE was not separating the instances from the different clusters. We see in Figures 2a and 2c that the projections from both known and unknown overlap for the most part. Hence, it is difficult to discriminate them. On the other hand, as shown in Figures 2b and 2d, ii-loss pushes the projections of different clusters further apart; so when an instance from unknown data distribution is encountered, the network projects them in the space between the clusters.

To understand the effect of the number of clusters (the dimension of the y-layer in Figure 1) and dimension of the z_ii layer (we will refer to as z_ii-dim) on the open set performance, we conducted 30 experiments on the three original datasets. We ran 30 experiments, similar to what we did in Section V-A, and present the results in terms of an average AUC. From Figures 3d, 3e, and 3f we observe that the change in the z_ii dimension has little effect on the open set performance. On the other hand, Figures 3a, 3b, and 3c show the effect of using a different number of clusters on the average AUC. We expected the open set recognition performance (i.e., average AUC) to increase as the number of clusters increases since the clusters become more homogeneous(pure) as the number of clusters increase. The results of MS and Android datasets seem to agree with our expectation as seen in Figures 3b and 3c. In case of MNIST dataset, however, average AUC decreases as the number of clusters increase, which is the opposite of what we expected to happen.

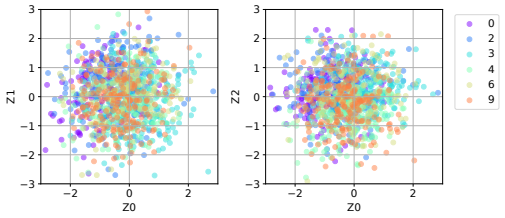
Our first hypothesis in trying to explain the unexpected result (i.e., the decrease in open set recognition performance with an increase in cluster number for MNIST dataset) was that it might be because the cluster quality(homogeneity) was not improving with an increase in the number of clusters. However, we found out that was not the case; cluster homogeneity in the experiments increases with an increase in the number of clusters. Our second hypothesis was that ii-loss was not separating the clusters well (hence explaining in poor open set recognition performance) when the number of clusters increased. We found out that inter-cluster separation did decrease as the number of clusters increased, however, this was the case for the MS and Android datasets as well, but in those two datasets, the open set recognition improved with an increase in the number of clusters. Therefore this does not explain these results. At this moment we are not able to find the answer to this unexpected result and leave this off as future work. The conclusion we can draw from these experiments is that the optimal cluster number (i.e., the dimension of the y-layer) for a good open set recognition performance differs from dataset to dataset and need to be tuned more carefully.

VII. Conclusion

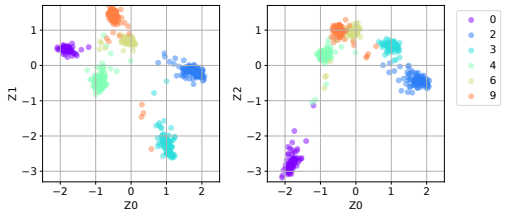
In this paper, we presented the problem of unsupervised open set recognition where the task is to identify instances in the test set that are not from the training data distributions. Our approach learns a representation for facilitating open set recognition in an unsupervised setting. We compare our pro-

TABLE II: Average F-score 30-runs.

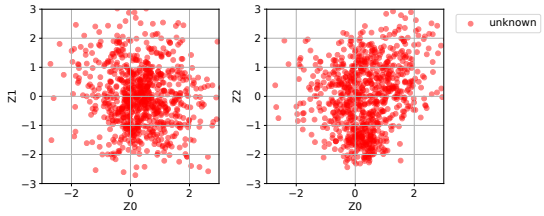
	Isolation Forest	KMeans-based	AAE-II
MNIST	0.39(±0.01)	0.49(±0.07)	**0.77**(±0.06)
MS	0.45(±0.15)	0.43(±0.10)	**0.65**(±0.10)
Android	0.29(±0.05)	0.44(±0.10)	**0.71**(±0.18)



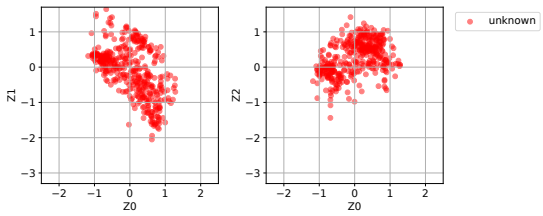
(a) z-layer projections
for instances from known data distribution.



(b) z_ii-layer projections
for instances from known data distribution.



(c) z-layer projections
for instances from unknown data distribution.



(d) z_ii-layer projections
for instances from unknown data distribution.

Fig. 2: Scatter plot of the six dimensional z-layer and z_ii-layer for 1000 random instances from known and unknown data distributions. The legend indicates the true label of the points.

TABLE III: 30-run average cluster homogeneity.

	AAE	AAE-II
MNIST	**0.834**(±0.049)	0.815(±0.076)
MS	0.771(±0.051)	**0.779**(±0.044)
Android	0.756(±0.040)	**0.757**(±0.040)

posed approach with closely related outlier detection methods and show that our approach gives a statistically significant improvement on three publicly available datasets. Two of the evaluation datasets are from the malware domain. To show the applicability of our approach to domains different from malware, we evaluate our approach on an image dataset as our third dataset. Our results also show that typical anomaly or outlier detection methods such as Isolation Forest do not perform when used unsupervised open set recognition tasks. Although the proposed approaches do perform better, there is still much room for improvement. For example, a more robust outlier threshold estimation approach can be explored.

REFERENCES

[1] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, "Toward open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1757–1772, 2013.

[2] H. Zhang and V. M. Patel, "Sparse representation-based open set recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 8, pp. 1690–1696, 2017.

[3] A. Bendale and T. E. Boult, "Towards open set deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1563–1572.

[4] W. J. Scheirer, L. P. Jain, and T. E. Boult, "Probability models for open set recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 11, pp. 2317–2324, 2014.

[5] H. Wang, X. Zhu, T. Xiang, and S. Gong, "Towards unsupervised open-set person re-identification," in *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 769–773.

[6] J. Kinable and O. Kostakis, "Malware classification based on call graph clustering," *Journal in computer virology*, vol. 7, no. 4, pp. 233–245, 2011.

[7] X. Hu, K. G. Shin, S. Bhatkar, and K. Griffin, "Mutantx-s: Scalable malware clustering based on static features." in *USENIX Annual Technical Conference*, 2013, pp. 187–198.

[8] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, 2011.

[9] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, behavior-based malware clustering." in *NDSS*, vol. 9. Citeseer, 2009, pp. 8–11.

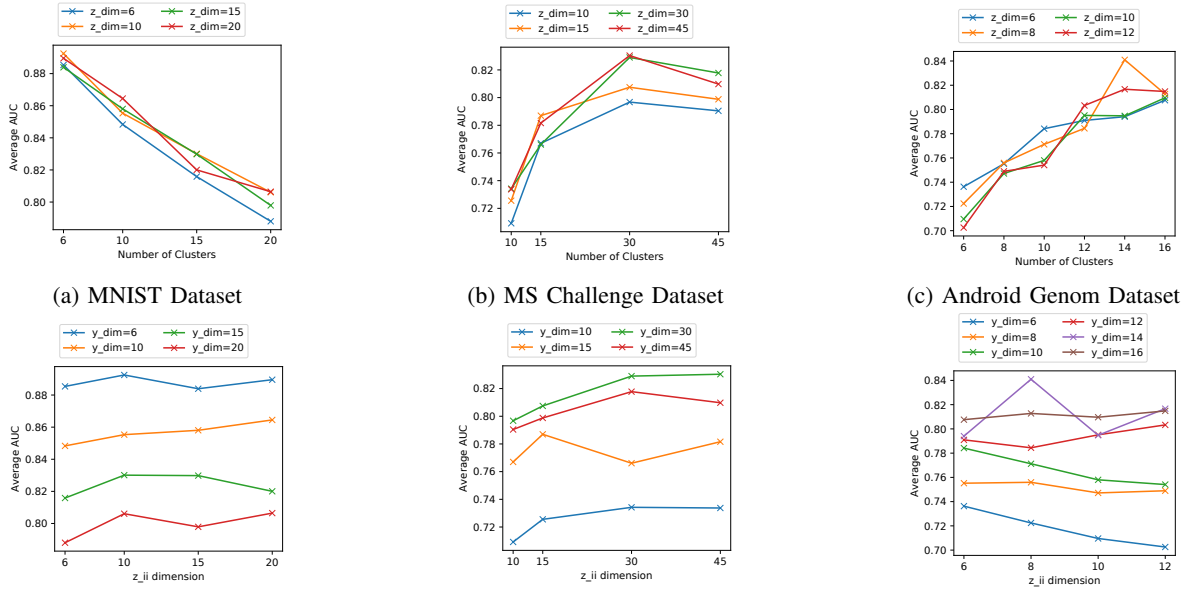
[10] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[11] M. Hassen and P. K. Chan, "Learning a neural-network-based representation for open set recognition," in *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 2020, pp. 154–162.

[12] T. G. Dietterich, "Steps toward robust artificial intelligence," *AI Magazine*, vol. 38, no. 3, pp. 3–24, 2017.

[13] L. P. Jain, W. J. Scheirer, and T. E. Boult, "Multi-Class Open Set Recognition Using Probability of Inclusion," pp. 393–409, 2014.

[14] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for



(a) MNIST Dataset (b) MS Challenge Dataset (c) Android Genom Dataset

(d) MNIST Dataset (e) MS Challenge Dataset (f) Android Genom Dataset

Fig. 3: Effect of cluster size and z dimension on performance.

mining outliers from large data sets," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 427–438.

[15] M. Sugiyama and K. Borgwardt, "Rapid distance-based outlier detection via sampling," in *Advances in Neural Information Processing Systems*, 2013, pp. 467–475.

[16] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM sigmod record*, vol. 29, no. 2. ACM, 2000, pp. 93–104.

[17] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 413–422.

[18] D. Yu, G. Sheikholeslami, and A. Zhang, "Findout: finding outliers in very large datasets," *Knowledge and Information Systems*, vol. 4, no. 4, pp. 387–412, 2002.

[19] "Microsoft malware classification challenge (big 2015)," https://www.kaggle.com/c/malware-classification, 2015, [Online; accessed 27-April-2015].

[20] "Android malware genome project," http://www.malgenomeproject.org/.

[21] M. Hassen and P. K. Chan, "Scalable function call graph-based malware classification." in *7th Conference on Data and Application Security and Privacy*. ACM, 2017, pp. 239–248.

[22] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007.