Representation Learning for Open Set Recognition and Novel Category Discovery

by

Jingyun Jia

A dissertation
submitted to the College of Engineering and Science
at Florida Institute of Technology
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Computer Science

Melbourne, Florida
December, 2022

We the undersigned committee
hereby approve the attached dissertation

Representation Learning for Open Set Recognition and Novel Category Discovery by Jingyun Jia

Philip Chan, Ph.D.
Associate Professor
Computer Engineering and Sciences
Major Advisor

Georgios Anagnostopoulos, Ph.D.
Associate Professor
Computer Engineering and Sciences

Debasis Mitra, Ph.D.
Professor
Computer Engineering and Sciences

Marius C. Silaghi, Ph.D.
Professor
Computer Engineering and Sciences

Philip J. Bernhard, Ph.D.
Associate Professor and Department Head
Computer Engineering and Sciences

ABSTRACT

Title:

Representation Learning for Open Set Recognition and Novel Category Discovery

Author:

Jingyun Jia

Major Advisor:

Philip Chan, Ph.D.

As machine learning models have achieved great success in various research and industry fields, the success of these models heavily relies on the massive amount of data collection and human annotations. While the real world is an open set, the daily emerged categories and the lacking of annotations have become new challenges for machine learning models. The absence of newly emerged categories in training samples can be captured by Open Set Recognition (OSR). Then, given the newly emerged samples, the process of automatically identifying the novel categories is called Novel Category Discovery (NCD). In this dissertation, we focused on learning the representations for OSR and NCD. To learn the representations for OSR, we first introduce an extension called Min Max Feature (MMF) that can be incorporated into different loss functions to find more discriminative representations. Our evaluation shows that the proposed extension can significantly improve the OSR performances of different types of loss functions. Then, we propose a self-supervision method Detransformation Autoencoder (DTAE), for the OSR problem in the image dataset. This proposed method engages in learning representations that are invariant to the transformations of the input data. Next, to extend DTAE to the graph dataset, we present two transformations (FCG-shift and FCG-random) for the Function Call Graph (FCG) based malware representations to facilitate the pretext task. The experiment results indicate that our proposed pre-training process can improve different performances of different downstream loss functions for the OSR problem in both image and graph datasets. To tackle the problem of NCD under an open-set scenario, we propose General Intra-Inter (GII) loss to learn a representation space that clusters the unlabeled samples as novel categories, meanwhile maintaining sensitivity to the unknown category. Our evaluation of image and graph datasets shows that GII outperforms other approaches in NCD and OSR.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I am extremely grateful to my advisor, Dr. Philip K. Chan, for giving me the opportunity to work on this topic, providing continuous academic support as well as invaluable research guidance, and challenging me to grow as a researcher. I would also like to thank my committee members for their expertise and precious time along the way.

I gratefully acknowledge the funding received towards this research from Amazon and the hardware received from Rockwell Collins. This dissertation would not have been possible without them.

Last but not least, my appreciation goes out to my family and friends for their encouragement and support throughout this adventure.

Dedication

I dedicate my dissertation work to my parents, Hailing Zhang and Hansheng Jia, for raising me up, instilling in me the value of being strong and independent, understanding and supporting all my choices.

I give my special thanks to my beloved partner, Nima Aghli, for being there, helping me every step of the way, and making this journey so special.

Chapter 1

Introduction

As machine learning techniques have succeeded greatly in various research and industry fields, most traditional classification problems focus on labeled samples. They are still facing various challenges in real-world applications. First, it is less likely to collect the training samples that exhaust all classes. Second, it is difficult and time-consuming to label all the classes in training samples. While more and more research topics have risen to meet these challenges, Geng et al. stated four categories of classes in the recognition problems as follows [25]:

- known known classes: labeled classes, available in training samples.
- known unknown classes: unlabeled classes, available in training samples.
- unknown known classes: training samples are not available, but some side-information, such as semantic/attribute information, is available.
- unknown unknown classes: neither training samples nor side-information

is available, completely unseen.

The problem of recognizing the classes that are not available in the training samples is referred as Open Set Recognition (OSR) [4]. That is, OSR attempts to handle “unknown unknowns classes” for more robust AI systems [18]. Hence, for a multinomial classification problem, an OSR task typically involves two objectives: to classify the known classes and reject the unknown class. The two objectives of OSR help build a more robust system than a traditional classifier. Such system defines a more realistic scenario and benefits the applications like face recognition [68], malware classification [41], and medical diagnoses [79].

After the unknown class is present and identified, the original “unknown unknown classes” become “known unknown classes”, and we face the problem of Novel Category Discovery (NCD). As the following step of OSR, NCD assumes two sets of samples with disjoint classes - the set of labeled samples from the “known known classes” and the set of unlabeled samples from the “known unknown classes”. The goal of NCD is to correctly classify the labeled samples meanwhile recognize the novel categories from the unlabeled samples without any annotation. Similar to the transitional classification problem, most research works in NCD assume a “close set” scenario. However, an ideal AI system should be able to maintain its robustness against an “open set” while discovering the novel classes.

Figure 1.1 shows an example of OSR and its following step of NCD. At stage 1, we labeled “dogs” and “cats” in the training set. In addition to “dogs” and “cats”, the test set contains “chickens” and “ducks”. An ideal OSR system should not only correctly classify the “dogs” and cats” but also recognize “chickens” and “ducks” as the unknown class, as shown in stage

2. Then we use the results in stage 2 as the training inputs of the NCD system. The NCD system should classify “dogs” and “cats”, meanwhile discovering two novel categories (“cluster 1” and “cluster 2”) as shown in stage 3. After labeling all four classes, we proceed to the next cycle of OSR (stage 4). Like stage 2, the ideal OSR system should correctly classify the four labeled classes while recognizing the unknown samples (“sheep” and “cows”). The continuous collaboration of the OSR and NCD systems provides a sustainable solution to the open-world problem. In this work, we propose several approaches for the OSR in stage 2 and a one-step solution for the NCD in stage 3 and OSR in stage 4.

1.1 Problem Statement

Representation learning or feature learning has proven to be effective in classification problems. A good representation can capture the underlying semantic distributions of the input samples and is useful as input to a supervised classifier or predictor. While most representation learning approaches capture the discrimination between classes with class labels, self-supervised learning extracts the representations of the input samples without the class labels. Unlike traditional supervised learning tasks, self-supervised learning uses a pretext task different from the primary task to learn the representations. A learning objective is set to get supervision from the training samples themselves.

As there are various ways to learn representations, we focus on neural network-based methods to extract the representations in this dissertation. Particularly, we are interested in three problems:

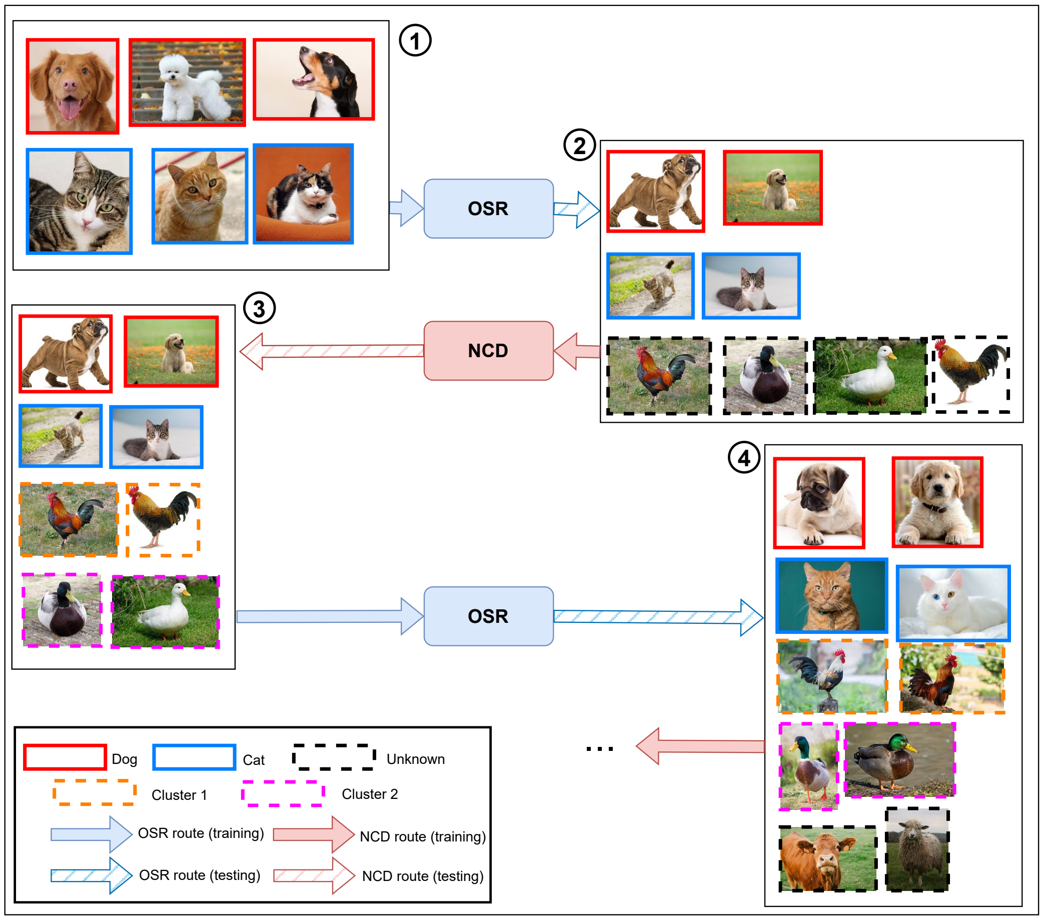


Figure 1.1: An example of Open Set Recognition (OSR) and Novel Category Discovery (NCD)

- Problem 1: enhancing the representation space that leverages intra-class spread and inter-class separation for OSR.
- Problem 2: initializing a representation space with self-supervised learning for OSR.
- Problem 3: learning a representation space that clusters the unknown samples and separates them from the known classes as a following step of OSR.

1.2 Approach

We propose a Min Max Feature (MMF) loss extension for Problem 1. MMF loss extension aims to help the existing loss functions handle the open set scenario better. The MMF loss extension emphasizes the features with the smallest and largest magnitudes during network training. The samples from unknown classes, which are not available in training, are not emphasized and remain small magnitudes. Thus, the known and unknown classes become more separable in the learned representation space during the inference.

We then follow a two-stage training process for the OSR problems for image data: A pre-training stage and a fine-tuning stage for Problem 2. During the pre-training stage, we propose a self-supervised learning approach, De-transformation Autoencoder (DTAE), to learn the low-level representations of the known classes without class labels. Like most self-supervised learning approaches, DTAE transforms the input samples into different views to facilitate the pretext task. It reconstructs the original input samples from these transformed views so that the learned representations are invariant to the transformations.

To extend DTAE to malware data, we analyze the characteristics of the Function Call Graphs (FCGs) from the malware datasets and propose two transformation methods: FCG-random and FCG-shift for the malware datasets. An FCG is a weighted directed graph, where the nodes represent the function clusters and the edges are the caller-callee relations between the function clusters. Our proposed transformation methods alter the orders of the function clusters. At the same time, maintain the caller-callee relations between the function clusters such that the transformed views are isomorphic graphs of the original input FCG samples.

The majority of self-supervised learning approaches, including DTAE, involve transformations. Thus, the learned representations include content features and transformation features. As the transformation information is usually ineffective in the downstream tasks, we propose Feature Decoupling (FD) to separate the content and transformation features. The content features are learned by DTAE. In addition, FD uses an auxiliary transformation classifier to learn the transformation features. In the second stage, the transformation features are discarded. The content features are further fine-tuned by class labels (supervised scenario) or directly clustered for OSR (unsupervised scenario).

After recognizing the unknown samples, we propose General Inter-Intra (GII) loss for Problem 3. GII aims to solve the NCD and OSR problems together. GII is designed to discover the novel classes from the unknown samples while separating them from the known classes in the representation space. GII uses K-means to find the representations' centroids of the unknown samples and sharpen the probability distribution of cluster assignment by decreasing weighted intra-cluster distances. Moreover, to accurately classify the known samples and separate the labeled known samples from the unknown ones, GII decreases intra-class distances for the labeled known samples and increases the distances between any two clusters or classes' centroids.

1.3 Contributions

Our contributions include the following:

- Our proposed MMF loss extension statistically significantly improved the OSR performance of different loss functions for image and malware datasets.
- Our proposed DTAE pre-training method can capture the cluster information for known and unknown samples even without class labels. Moreover, it boosts the OSR performance for different downstream loss functions on several image datasets.
- We propose FCG-shift and FCG-random transformations based on the characteristics of the malware FCGs. These transformation methods facilitate the self-supervised pre-training process and improve the model performance for the malware OSR tasks.
- We propose FD to extract the content features in the self-supervised pre-training process. FD is more effective than other approaches in supervised and unsupervised OSR tasks.
- We propose GII to discover the novel classes in the unknown samples while maintaining the open set robustness of the system. GII outperforms other approaches in NCD and OSR tasks.

1.4 Overview

We provide a broad review of related work in Chapter 2. The review covers three major topics of this dissertation: Open Set Recognition, Self-supervised Learning, and Novel Category Discovery.

We then present our approaches to OSR and NCD tasks from Chapter 3 to Chapter 7. Specifically, In Chapter 3, we propose MMF loss extension and incorporate it into different loss functions. In Chapter 4, we introduce a two-stage training process for the OSR tasks. We also propose DTAE as a self-supervised pre-training method. In Chapter 5, we summarize the characteristics of the malware FCGs and propose two transformation methods for the malware FCGs to facilitate the self-supervised pre-training process for the OSR tasks. In Chapter 6, we propose a feature decoupling approach to extract the content features and extend our approach to the unsupervised scenario in OSR. Moreover, we propose an intra-inter ratio (IIR) metric for OSR performance. In Chapter 7, we propose GII to discover the novel classes in the unknown samples as the following step of the OSR. We evaluate GII for both NCD and OSR performances.

Finally, we present a summary of our approaches in Chapter 8. We also analyze the limitations of our work and discuss future research directions.

Chapter 2

Related Work

In this chapter, we summarize the related work in four topics: representation learning, self-supervised learning, open set recognition (OSR) and novel category discovery (NCD). First, in section 2.1, we give a brief introduction to neural network-based representation learning methods as well as associated representation loss functions. Then, we discuss self-supervised learning methods in images and graphs domains separately in section 2.2. Next, section 2.3 divides current OSR techniques into three categories based on the training set components: training with borrowed additional data, training with generated additional data, and training without additional data. Finally, we give brief descriptions of neural network-based NCD methods in section 2.4.

2.1 Representation Learning

Representation learning, also known as feature learning, is the process of learning data representations that make it easier to extract useful information

Table 2.1: Representation loss functions.

Loss functions	Utilities
Triplet loss [80]	Introduced for supervised face recognition tasks
N-pair loss [86]	Solve slow convergence for triplet loss
Center loss [92]	Introduced for supervised face recognition tasks
Triplet-center loss [42]	Combine triplet loss and center loss
Ii loss [41]	Proposed for open set recognition

when building classifiers or other predictors [5]. It allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification [57]. Those raw data could be images, graphs, texts, etc. An image comes in the form of an array of pixel values, and texts come in the form of word sequences. Motivated by different objectives, a set of representative features would be generated through deep neural networks. We call this type of objectives representation loss.

The objective of representation loss functions is to learn better representations of training data, and they are normally applied to the representation layers. Table 2.1 shows some examples of representation loss functions along with their utilities. They are triplet loss [80], N-pair loss [86], center loss [92], triplet-center loss [42] and ii loss [41].

Triplet loss [80] 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 (positive example) is smaller by a user-specified margin than the distance between the anchor instance and another instance from a different class (negative example).

As triplet loss employs only one negative sample while not interacting with other negative classes in each update, it suffers from slow convergence. N-pair loss [86] addresses this problem by generalizing the objective to multiple negative samples. To alleviate the computation growth due to the additional negative samples, N-pair loss uses an efficient batch construction method requiring fewer examples to achieve the same objective.

Wen et al. propose center loss [92] for supervised face recognition. In addition to softmax loss. The center loss learns the representation centers of each class and penalizes the distances between the representations and their corresponding class centers. Training with softmax loss jointly, center loss effectively characterizes the intra-class variation.

Inspired by triplet loss and center loss, He et al.[92] propose triplet-center loss (TCL) to further enhance the discriminative power of the learned representations. TCL leverages the advantages of triplet loss and center loss to minimize the intra-class distance and maximize the inter-class distances of the representations. Center loss only considers intra-class variation and ignores the inter-class separation and triplet loss subjects to the complexity of the construction of triplets. TCL combines triplet loss and center loss. Compared to center loss, it also considers inter-class separation. Moreover, instead of comparing the distances of each two instances in triplet loss, TCL computes the distances of instance and class center, thus fewer triplets to be constructed.

Hassen and Chan [41] propose ii loss 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, in the learned representation, instances from the same class are close to each other while those from different classes are further apart.

2.1.1 Feature disentanglement and decoupling

Many research works focus on feature disentanglement and decoupling to achieve better generalization in representation learning. Feng et al. [20] propose rotation classification to extract rotation information. They also propose rotation irrelevance and image instance classification, which encourage feature vectors of the rotated images to be similar, but feature vectors representing different original images to be far apart. The authors point out that rotation irrelevance can produce degenerate solutions and needs additional regularization.

Peng et al. [70] propose learning a novel reconstruction-based identity feature representation invariant to pose. Specifically, they present a feature disentanglement approach for face recognition, where they introduce a feature reconstruction metric learning to disentangle identity features and pose features by demanding alignment between the feature reconstructions through various combinations of identity and pose features.

Moreover, Tang et al. [88] disentangle place and appearance features (domain) for place recognition. They propose to use a self-supervised adversarial network to disentangle domain-unrelated (place) and domain-related (appearance) features. Specifically, they apply adversarial training explicitly among place features. Moreover, another adversarial loss is used to eliminate dependency between place features and appearance features.

To estimate head pose from RGB images, Zhang et al. [103] present a three-branch network architecture named Feature Decoupling Network (FDN) for landmark-free head pose estimation from an image. They introduce a feature decoupling module to learn the discriminative features for each pose angle explicitly. Moreover, they propose a cross-category center

(CCC) loss to obtain more compact and distinct latent variable subspaces.

2.1.2 Learning representation for graphs

Graphs are essential data structures in machine learning tasks, and the challenge is to find a way to represent graphs. Traditionally, feature extraction relied on user-defined heuristics. Furthermore, recent research has focused on using deep learning to learn to encode graph structure into low-dimensional embedding automatically. Hamilton et al. [33] provided a review of representation learning techniques on graphs, including matrix factorization-based methods, random-walk-based algorithms, and graph networks. The paper introduced methods for node embedding and subgraph embedding. The node embedding can be viewed as encoding nodes into a latent space from an encoder-decoder perspective. Subgraph embedding aims to encode a set of nodes and edges, a continuous vector representation. Many of the methods build upon the techniques of node embedding.

2.1.2.1 Vertex embedding

Vertex embedding can be organized as an encoder-decoder framework. An encoder maps each node to a low-dimensional vector or embedding. And decoder decodes structural information about the graph from the learned embeddings. Several deep neural network-based approaches have been proposed to address the above issues. They used autoencoders to compress information about a node's local neighborhood.

Grover and Leskovec [31] propose node2vec to learn continuous feature representations for nodes in networks. To learn richer representations of nodes, they design a random walk procedure, which can efficiently explore the diverse neighborhoods of nodes in the networks. Moreover, node2vec maps the nodes to a low-dimensional representation space, maximizing the likelihood of preserving the network neighborhood of nodes.

In the work of Graph Convolutional Networks (GCN) [54], Kipf and Welling encode the graph structure directly using a neural network model and trained on a supervised target. The adjacency matrix of the graph will then allow the model to distribute gradient information from the supervised loss. It will enable it to learn representations of nodes with and without labels.

Hamilton et al. present GraphSAGE [34] as a general inductive framework that leverages node feature information to generate node embeddings for previously unseen data efficiently. GraphSAGE can learn a function that generates embeddings by sampling and aggregating features from a node's local neighborhood. Instead of training individual embedding for each node, a set of aggregator functions are learned to aggregate feature information from a node's local neighborhood from a different number of hops away from a given node. The learned aggregation functions are then applied to the entire unseen nodes to generate embeddings during the test phase.

Tang et al. proposed a method for Large-scale Information Network Embedding (LINE) in [87], which is suitable for undirected, directed and/or weighted networks. The model optimizes an objective that preserves local and global network structures. LINE explores both first-order and second-order proximity between the vertices. The functions that preserve first-order and second-order proximities are trained separately, and the embeddings trained by two methods are concatenated for each vertex.

To overcome the limitations of neighborhood aggregation schemes, Xu et al. proposed Jumping Knowledge (JK) Networks strategy in [95] that flexibly leverages different neighborhood ranges to enable better structure-aware representation for each node. This architecture selectively combines different aggregations at the last layer, i.e., the representations "jump" to the last layer. As a more general framework, JK-Net admits general layer-wise aggregation models and enables better structure-aware representations on graphs with complex structures.

Cao et al. [10] adopt a random surfing model DNGR to capture structural graph information directly instead of using a sampling-based method. DNGR contains three major components: random surfing, calculating the PPMI matrix, and feature reduction by SDAE. The random surfing model is motivated by the PageRank model and is used to capture graph structural information and generate a probabilistic co-occurrence matrix.

2.1.2.2 Subgraph embedding

The goal of embedding subgraphs is to encode a set of nodes and edges into a low-dimensional vector embedding. Representation learning on subgraphs is closely related to the design of graph kernels, which define a distance measure between subgraphs. According to [33], some subgraph embedding techniques use the convolutional neighborhood aggregation idea to generate embeddings for nodes then use additional modules to aggregate sets of node embeddings to subgraph, such as sum-based approaches, graph-coarsening approaches.

Gilmer et al. [28] propose Message Passing Neural Networks (MPNNs) to reformulate existing message passing algorithms and aggregation procedures into a single framework for chemical predictions. Mainly, MPNNs operate on undirected graphs with node features and edge features. The forward pass has a message passing phase and a readout phase. During the message passing phase, the hidden states of each node in the graph are updated based on the node itself and its neighbors. The readout phase computes a feature vector for the whole graph using some readout function.

Duvenaud et al. [19] introduce a convolutional neural network that operates directly on graphs for learning molecular fingerprints. In the graphs, vertices represent individual atoms, and edges represent bonds. The same local filter is applied to each atom, and its neighborhood in the convolutional blocks and a global pooling layer combines features from all the atoms in the molecule. This architecture generalizes molecule feature extraction methods based on circular fingerprints.

Battaglia et al. [2] introduce an interaction network to perform an analogous form of reasoning about objects and relations in complex systems. Interaction networks have three approaches: structured models, simulation, and deep learning. It takes graphs as input and performs object- and relation-centric reasoning via deep neural networks. This work considers the case where there is a node-level target and where there is a graph-level target. It also considered the case where node-level effects are applied at each time step.

Kearnes et al. [53] propose graph convolutions for molecular fingerprints encoding. Graph convolutions use a representation of small molecules as undirected graphs of atoms. The graph structure presents atom and bond properties, and the graph distances describe molecule-level representations.

Schütt et al. [83] propose a deep tensor neural network (DTNN) to enable spatially and chemically resolved insights into quantum-mechanical properties of molecular systems. In DTNN, the message is computed by matrices and bias vectors, and the readout function passes each node independently through a single hidden layer neural network and sums the outputs.

Moreover, Bruna et al. [8] and Defferrard et al. [16] introduce Laplacian based methods in learning graphs. These methods generalize the notion of the convolution operation typically applied to image datasets to an operation that operates on an arbitrary graph with a real-valued adjacency matrix. Particularly, in their message functions, the matrices are parameterized by the eigenvectors of the graph laplacian as well as the learned parameters of the model. Furthermore, the vertex update function has pointwise non-linearity (such as ReLU).

2.2 Self-supervised Learning

Unlike the traditional supervised learning approaches, which use human annotations to guide the training process for the primary tasks, self-supervised learning methods learn representations via pretext tasks that are different from the primary tasks. Recent works have applied self-supervised learning to various domains. In this section, we focus on self-supervised learning in image samples and graph samples.

2.2.1 Self-supervised Learning in Images

Autoencoding is a simple example of self-supervised learning. It learns the representation by reconstructing the original input samples. Denoising autoencoder (DAE) [90] corrupts the input samples first. The network is trained to denoise corrupted versions of their inputs to reconstruct them back to their original forms. Additionally, more research works introduce these pretext tasks by transforming the original inputs into different views.

For example, in the image representation learning, Gidaris et al. [27] propose RotNet, where the pretext task predicts the rotation degree (e.g., 0, 90, 180 and 270 degrees). To achieve better generalization ability, Feng et al. [20] decouples the rotation discrimination features from instance discrimination features. The rotation discrimination features are discovered by predicting image rotations, and the instance discrimination features are learned by penalizing the distance difference between features of the same image under different rotations. Chen et al. [12] propose SimCLR. It first generates positive and negative pairs between different views and proposes contrastive loss to increase the similarity between positive pairs meanwhile decrease the similarity between negative pairs. Zbontar et al. [102] propose Barlow Twins, which generates the cross-correlation matrix between the representations of two views of the same input. Then, it tries to make this matrix close to the identity to reduce the redundancy in learned features. As the siamese network has become a common structure for self-supervised learning in images, Chen and He propose SimSiam [13], which does not need negative pairs in the training process. Moreover, SimSiam uses a stop-gradient operation to prevent collapsing solutions.

2.2.2 Self-supervised Learning in Graphs

Besides image representation learning, more recent works have extended self-supervised learning to graph representation learning. You et al. [100] propose Graph contrastive learning (GraphCL), which extends the contrastive learning framework in SimCLR to the graph field. Specifically, it designs four types of transformations for the graph inputs: node dropping, edge perturbation, attribute masking, and subgraph sampling. Jin et al. [51] propose Pairwise Half-graph Discrimination (PHD). PHD first generates two augmented views based on local and global perspectives from the input graph. Then, the objective function maximizes the agreement between node representations across different views and networks. While there are various graph transformation methods and graph learning tasks, Xu et al. [94] propose InfoGCL to find the optimal approach for particular graph learning tasks and datasets. They break down graph contrastive learning approaches into three stages and utilize an Information Bottleneck (IB) information-aware framework to find the optimal modules in each stage.

2.3 Open Set Recognition (OSR)

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. We can divide OSR techniques into three categories based on the training set compositions. The first category borrows additional data from other datasets and uses them as the unknown class in the training set. Thus the N-class classification problem becomes (N+1)-class classification problem. The second category generates additional data and uses them as the unknown class in the training set. Instead of converting the N-class classification problem to the (N+1)-class classification problem, the third category does not require additional data in the training set.

2.3.1 Training with borrowed additional data

The first category includes the techniques that borrow additional data in the training set. Saito et al. [77] propose a method that marks unlabeled target samples as unknown, then mixes them with labeled source samples to train a feature generator and a classifier. The classifier attempts to between source and target samples, whereas the generator attempts to make target samples far from the boundary. The idea is to extract features that separate known and unknown samples. According to the feature generator, the test data either would be aligned to known classes or rejected as an unknown class.

Shu et al. [84] introduce a framework to solve the open set problem, which involves unlabeled data as an autoencoder network to avoid overfitting. Besides the autoencoder, it contains another two networks in the training process - an Open Classification Network (OCN) and a Pairwise Classification Network (PCN). Only OCN participants are in the testing phase, which predicts the test dataset, including unlabeled examples from both seen and unseen classes. Then it follows the clustering phase. Based on the prediction results of PCN, they used hierarchical clustering (bottom-up/ merge) to cluster rejected examples clusters.

Moreover, Dhamija et al. [17] utilize the differences in feature magnitudes between known and borrowed unknown samples as part of the objective function. Shu et al. [85] indicate that several manual annotations for unknown classes are required in their workflow. Hendrycks et al. [43] 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, retrieving and selecting additional datasets remains an issue.

2.3.2 Training with generated additional data

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 Generative adversarial networks (GANs).

Ge et al. [24] design a networks based on OpenMax and GANs. Their approach provided explicit modeling and decision score for novel category image synthesis. The method proposed has two stages and OpenMax: pre-Network training and score calibration. The pre-Network training stage, different from OpenMax, generates some unknown class samples (synthetic samples) and sends them with known samples into networks for training. For each generated sample, if the class with the highest value differs from the pre-trained classifier, it will be marked as "unknown". Finally, a final classifier provides an explicit estimated probability for generated unknown unknown classes.

Yu et al. [101] propose Adversarial Sample Generation (ASG) as a data augmentation technique for OSR problem. The idea is to generate some points close to but different from the training instances as unknown labels, then straightforward to train an open-category classifier to identify seen from unseen. Moreover, ASG also generates "unknown" samples, which are close to "known" samples. Unlike GANs min-max strategy, ASG generated samples based on distances and distributions. The generated unknown samples are close to the seen class data and scattered around the known/unknown boundary.

Furthermore, Neal et al. [66] add another encoder network to traditional GANs to map from images to a latent space. Jo et al. [52] generate unknown samples by a marginal denoising autoencoder that provides a target distribution away from known samples' distribution. Lee et al. [59] 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.

2.3.3 Training without additional data

The third category of open set recognition does not require additional data. Most of the research works require outlier detection for the unknown class.

Bendale and Boult[4] propose OpenMax, which replaces the softmax layer in DNNs with an OpenMax layer, and the model estimates the probability of an input being from an unknown class. The model adapts Extreme Value Theory (EVT) meta-recognition calibration in the penultimate layer of the networks. For each instance, the activation vector is revised to the sum of the product of its distance to the mean activation vectors (MAV) of each class. Further, it redistributes values of activation vector acting as activation for unknown unknown class. Finally, the new redistributed activation vectors are used for computing the probabilities of both known and unknown classes.

Schultheiss et al. [81] investigate class-specific activation patterns to leverage CNNs to novelty detection tasks. They introduced Extreme Value Signature (EVS), which specifies which dimensions of deep neural activations have the largest value. They also assumed that a semantic category could be described by its signature. Thereby, a test example will be considered novel if it is different from the extreme-value signatures of all known categories. They applied extreme value signatures on top of existing models, allowing them to "upgrade" arbitrary classification networks to estimate novelty and class membership jointly.

Additionally, Pidhorskyi et al. [72] propose manifold learning based on training an Adversarial Autoencoder (AAE) to capture the underlying structure of the distributions of known classes. Hassen and Chan [41] 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. [91] design a distance-based loss function with a user-specified margin between classes named pairwise loss to separate different classes.

2.4 Novel Category Discovery (NCD)

After identifying the unknown samples from the OSR, the next step is to discover the novel categories from the unknown samples. In the real world, the unknown samples are usually a mixture of several classes. As we do not know the labels of the unknown instances, how to separate these unknown classes becomes an unsupervised clustering problem.

Most current researches focus on transforming the clustering problem into pairwise similarity prediction. Gupta et al. [32] utilize the Information Maximization (IM) loss in an ensemble of models to predict the similarity between two data points. The training objective is to maximize the mutual information between the input and the model output while imposing some regularisation penalty on the model parameters. However, the IM loss assumes the dataset is balanced and performs poorly on an imbalanced dataset.

Han et al. [36] assume prior knowledge of related but different unlabeled image classes. They use such prior knowledge to reduce the ambiguity of clustering by reducing its KL divergence to a sharper target distribution.

They also introduce a method to estimate the number of unlabeled data by transferring the knowledge of labeled ones. They separated the known classes into three subsets, including one used for pre-training and the other two subsets combined with the unlabeled data for class number estimation. Specifically, one subset is used for constrained k means, and the other is used for evaluating the clustering performance. Finally, they select the k that achieves the best clustering performance.

Zhong et al. [106] introduce OpenMix to mix the unlabeled examples from an open set and the labeled examples from known classes. They follow a two-stage learning stage for the NCD problem. The model initialization stage is trained on the labeled samples in a supervised way. In the unsupervised clustering stage, they generate mixed training samples by incorporating labeled samples with unlabeled samples, the pseudo-labels of mixed samples will be more reliable than their unlabeled counterparts. In addition to pseudo-pair learning and pseudo-label learning, the loss of OpenMix is applied to the mixed samples.

Vaze et al. [89] consider a Generalized Category Discovery (GCD) setting for image recognition, where the unlabelled samples may come from labeled classes or novel ones. They propose using vision transformers with contrastive representation learning and semi-supervised k-means for such a setting. Specifically, their approach has three steps. The first step uses a vision transformer to learn the low-level representations of the input samples in a self-supervised manner. Then, they assign the labels of unlabelled data using semi-supervised k-means clustering. In the last step, they estimate the class numbers by evaluating the clustering accuracy on the labeled samples.

Fini et al. [21] introduce a UNified Objective function (UNO) for discovering novel classes, they eliminate the self-supervised pre-training step and combine supervised and supervised learning in a single framework. They transform the training sample into two views, assign pseudo-labels to the samples, then the network predicts a probability distribution over all classes for each view. The cluster assignment of each view is used as the pseudo-label for the other view. In UNO, the cluster pseudo-labels are treated homogeneously with ground truth labels, and a cross-entropy loss is applied on both labeled and unlabeled samples.

In addition, Chang et al. [11] propose DAC architecture, which uses the learned label features for clustering tasks via introducing a constraint. The sample pairs used for training are alternately selected and labeled by the learned features in each iteration. Likewise, Hsu et al. [44] propose Meta Classification Learning (MCL), which uses the learned label features to measure the similarity of a sample pair. They use a multi-class classifier as a submodule, which generates label features for calculating similarity. Optimizing a binary classifier on top of it allows the learned multi-class classifier to be learned and used to predict the unknown class in the test phase. Liu et al. [64] propose ResTune to estimate a new residual feature from the pre-trained network and add it with a previous basic feature to compute the clustering objective. ResNet consists of three components: a clustering objective to cluster the unlabeled images, a knowledge distillation objective to overcome catastrophic forgetting, and a pairwise labeling objective to preserve the structure information. Yang et al. [98] propose to divide and conquer the generalized settings of the NCD problem with two groups of Compositional Experts (ComEx): batch-wise experts and class-wise experts. The batch-wise expert is implemented as a linear classifier to handle the disjoint part of the dataset. Furthermore, the class-wise experts include the base-class expert for the labeled samples and the novel-class expert to handle the novel classes. Jia et al. [50] present an end-to-end framework to discover categories in unlabelled data. In addition to supervised learning on labeled samples, they apply contrastive learning on both labeled and unlabeled samples. Moreover, they employ the Winner-Take-All (WTA) hashing [96] on the shared representation space to generate pseudo-labels for the unlabeled samples in the training process. Zhong et al. [105] propose Neighborhood Contrastive Learning (NCL) framework to address the NCD problem. NCD uses constructive learning in the training samples. It considers the local neighborhood of a sample in the representation space as pseudo-positives and generates hard negatives by mixing labeled and unlabeled samples in the representation space. Zhao and Han [104] propose to apply dual ranking statistics to transfer the knowledge learned from labeled samples to unlabelled samples for pseudo-labeling. In addtion, they present a two-branch learning framework to tackle the problem of NCD. One branch focuses on local part-level information, while the other focuses on global characteristics. Meanwhile, a mutual knowledge distillation method is introduced to encourage agreement on the local and the global branches.

2.5 Malware Classification

Malware classifiers often use sparse binary features, and there can be hundreds of millions of potential features. Dahl et al. [15] first extract three types of features, including null-terminated patterns observed in the process' memory, tri-grams of system API calls, and distinct combinations of a single system API call and one input parameter. To reduce the dimensionality of the features, Dahl et al. use random projections to reduce the dimensionality of the original input space of neural networks, and the lower-dimensional data serves as input to the neural network.

Hassen and Chan [38] use a linear time function call graphs (FCGs) to extract malware representations for classification. The proposed malware classification system starts with an FCG extraction module, which is a directed graph representation of code where the vertices of the graph correspond to functions and the directed edges represent the caller-callee relation between the function nodes. The FCGs have a good performance in saving the interaction information between functions. The FCG extraction module takes disassembled malware binaries and presents the caller-callee relation between functions as directed, unweighted edges. Then a function clustering module uses minhash to cluster functions of the given graph. Next, a vector extraction module extracts vector representation from the FCGs. The representation consists of two parts, vertex weight and edge weight. The vertex weight specifies the number of vertices in each cluster for that FCG, and the edge weight describes the number of times an edge is found from one cluster to another cluster. Based on this work, Hassen and Chan further introduce two new features in Classification in an Open World (COW) [40]: the maximum predicted class probability for one instance and the entropy for probability distribution over classes for malware classification.

Chapter 3

MMF: A Loss Extension for Feature Learning in Open Set Recognition

3.1 Introduction

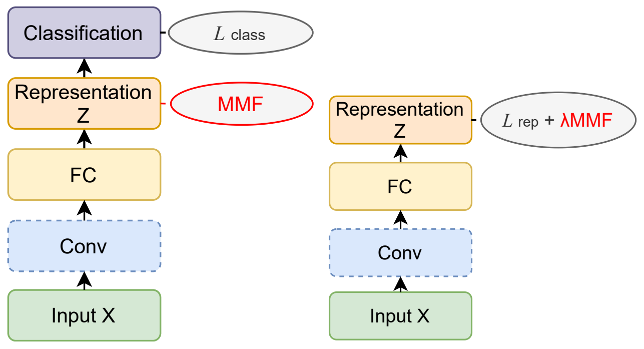
The OSR problem aims to classify the multiple known classes for a multinomial classification problem while identifying the unknown classes. The OSR problem defines a more realistic scenario and has drawn significant attention in application areas.

In this chapter, 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 [81]. 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 chapter 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 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 [41] outperforms the other combinations in both training time and overall F1 score.

We organize the chapter as follows. Section 3.2 presents the MMF loss extension. Finally, section 3.3 shows that the MMF extension can improve different types of loss functions significantly.

3.2 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.



(a) With classification loss (b) With representation loss

Figure 3.1: 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.

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. For example, the network in Figure 3.1a 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 3.1b, where we use learned representations instead of a classification layer for classification tasks. Figure 3.3a 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

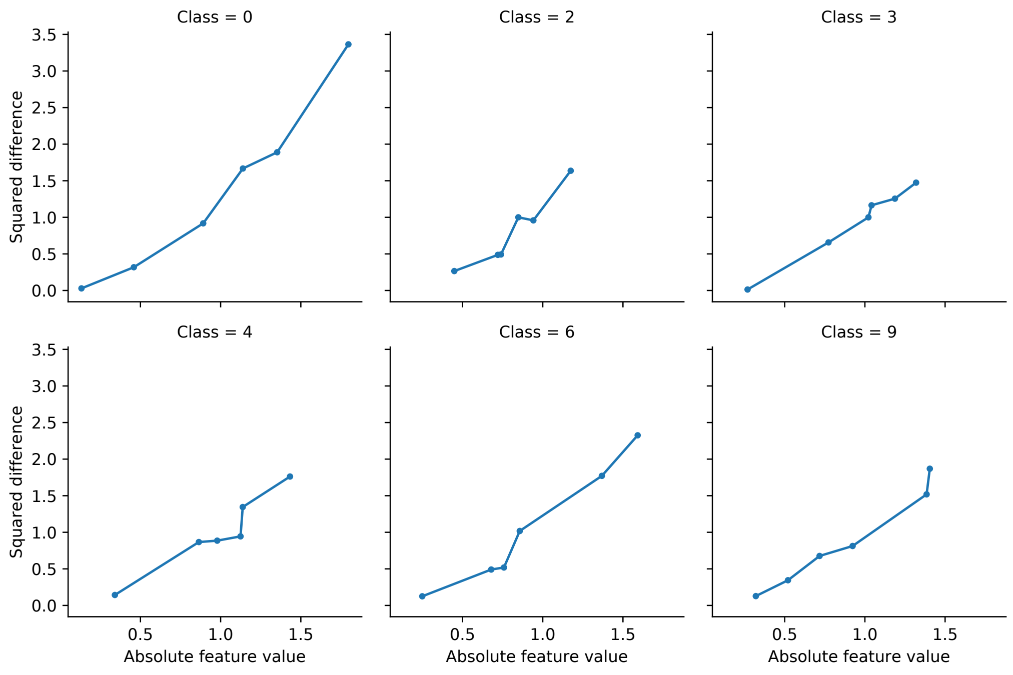
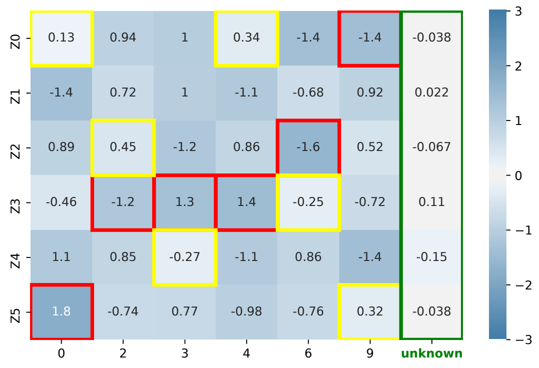
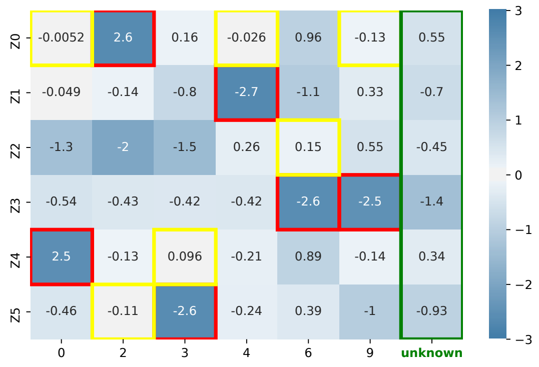


Figure 3.2: Squared differences of MAV values between the known and unknown classes in Figure 3.3a. The x-axis is the absolute feature values in six features, and the y-axis is their corresponding squared differences to the unknown class.

components of Euclidean distance. Thus we can increase the distance by increasing squared differences. Figure 3.2 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 3.3a. 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


(a) Standalone ce


(b) With MaxF

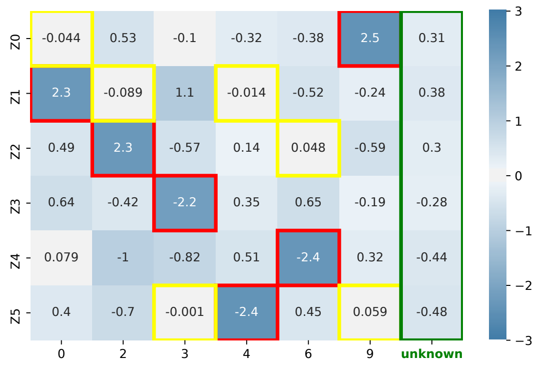

(c) With MMF

Figure 3.3: 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.

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 3.3b, the MAV of the unknown class (green column) has significantly increased compared to the one only using standalone cross-entropy loss in Figure 3.3a. 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 3.3c and Figure 3.3a, 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 3.3a is 2.32. After adding "MMF" in 3.3c 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.

3.2.1 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:

$$max\_feature_i = \max_{1\leq j\leq F} |\mu_{ij}|, \tag{3.1}$$

In the example of Figure 3.3a, 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:

$$min\_feature_i = \min_{1\leq j\leq F} |\mu_{ij}| \tag{3.2}$$

The "min_feature" in the example of Figure 3.3a 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:

$$MMF = \max_{1 \leq i \leq C}(min\_feature_i) - \min_{1 \leq i \leq C}(max\_feature_i) \tag{3.3}$$

In the example of Figure 3.3a, 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.

3.2.2 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 3.1.

We use the network architecture in Figure 3.1a to simultaneously train the network with classification loss functions and the MMF extension. During each iteration, we first extract AVs and generate the representation matrix as shown from line 2 and line 5 in Algorithm 1, then we construct the MMF extension function from the "max_feature" vector (max_f) and "min_feature" vector (min_f) as shown from line 6 to line 8. 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 15.

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
 1: for number of iterations do
 2:     $\boldsymbol{z} \leftarrow$ Extract AVs from the penultimate layer
 3:     for i = 1...C do
 4:         $\mu_i \leftarrow mean_j(\boldsymbol{z})$                    ▷ MAVs for each class
 5:     $\mathbb{U} \leftarrow STACK(\mu_i)$
 6:     $max\_f \leftarrow get\_max\_feature(\mathbb{U})$                ▷ Eq. 3.1
 7:     $min\_f \leftarrow get\_min\_feature(\mathbb{U})$                ▷ Eq. 3.2
 8:     $MMF \leftarrow \max(min\_f) - \min(max\_f)$                    ▷ Eq. 3.3
 9:     if representation loss function $\mathcal{L}_{rep}$ then
10:         $\mathcal{L} \leftarrow \mathcal{L}_{rep} + \lambda MMF$
11:     else
12:         $\mathcal{L} \leftarrow MMF$
13:         $\boldsymbol{W} \leftarrow SGD(\boldsymbol{W}, \mathcal{L})$
14:     if classification loss function $\mathcal{L}_{class}$ then
15:         $\boldsymbol{W} \leftarrow SGD(\boldsymbol{W}, \mathcal{L}_{class})$
    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:

$$\mathcal{L} = \mathcal{L}_{rep} + \lambda MMF, \tag{3.4}$$

$\mathcal{L}_{rep}$ is a representation loss function, and λ is a hyperparameter that strikes a balance between the representation loss function and the MMF extension. Figure 3.1b 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 [41]. 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.

3.3 Experimental Evaluation

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

MNIST [74] 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 [56] 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) [58] 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 [39] 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) [108] 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 [23] to extract the function instructions and then extract 1453 raw FCG features as in [39]. 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.

3.3.1 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 [41] and triplet loss [80] as the examples of representation loss functions. Moreover, we compare these pairs with OpenMax [4].

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 rate 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 set the Dropout rate as 0.1. 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 3.2.2, we use contamination ratio of 0.01 for the threshold selection.

As we discussed in Equation 3.4, 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

Table 3.1: 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

(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.

3.3.2 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 3.1 shows

Table 3.2: 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

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).

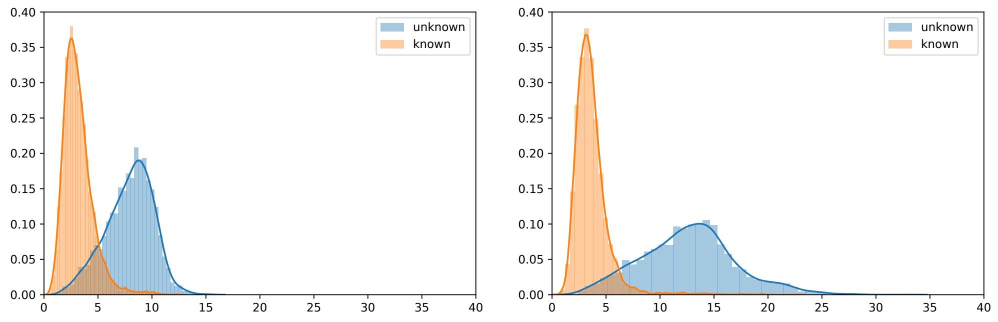
Table 3.2 shows the average F1 scores for the four datasets. We first calculate the F1 scores for each of the C known classes and the unknown class, then average the $C + 1$ classes as the Overall F1 scores. We can see that the loss functions with the MMF extension have better results than their corresponding 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 3.3 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.

3.3.3 Analysis

Figure 3.3c 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 3.3a), 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 3.4 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



(a) triplet (b) triplet+MMF

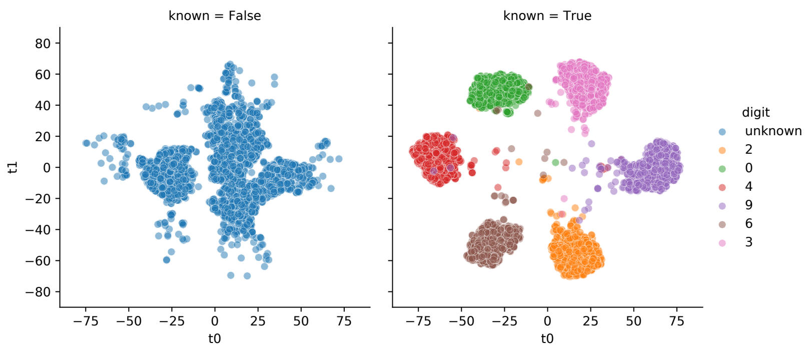
Figure 3.4: The distributions of outlier scores in MNIST.

Table 3.3: The comparison of training time for the MNIST dataset.

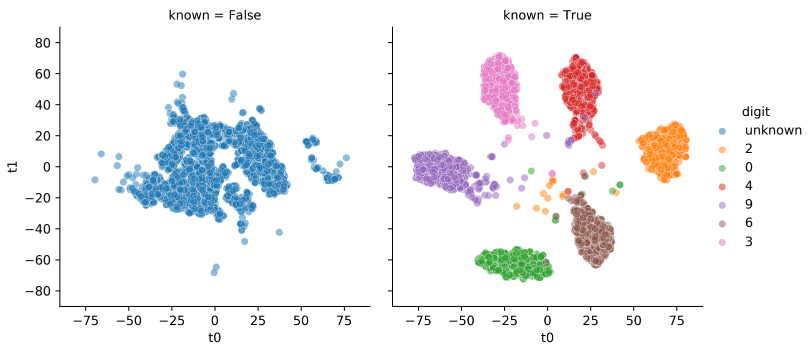
	Regular	+MMF	delta
ce	119.33	230.43	+111.1
ii	122.17	123.30	+1.14
triplet	223.27	225.70	+2.43

and the unknown classes more separable than before. Figure 3.5 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 3.1, 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 3.2.

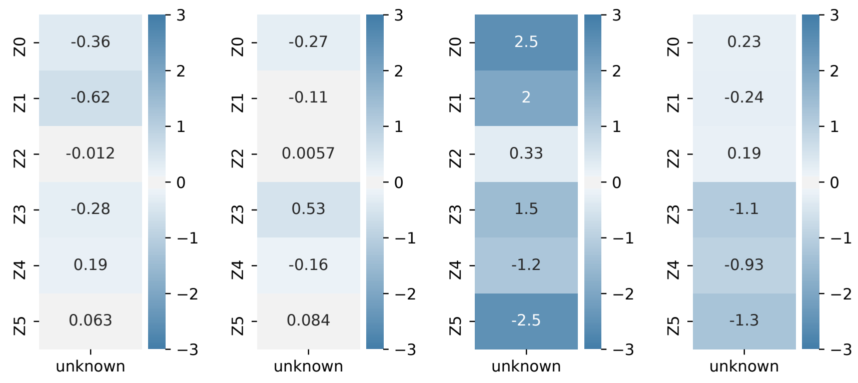


(a) triplet



(b) triplet+MMF

Figure 3.5: 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

Figure 3.6: The heatmap of the unknown class's MAV in the experiment of cross entropy loss (ce) on the Microsoft Challenge dataset (MC).

Moreover, we find that MaxF and MinF extensions can also achieve better performance than standalone loss functions. 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 3.6. Comparing Figure 3.6a and Figure 3.6b, 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 3.6c and Figure 3.6d, 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 3.2. In addition, we observed similar behaviors from other datasets.

3.4 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.

Chapter 4

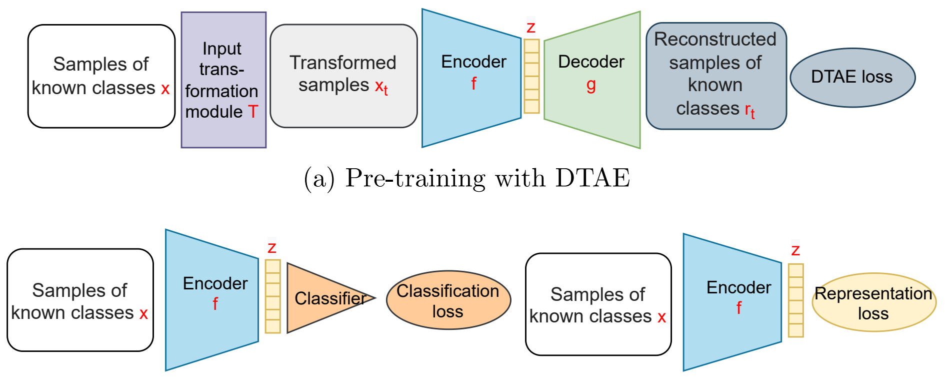
Self-supervised Detransformation Autoencoder for Representation Learning in Open Set Recognition

4.1 Introduction

Deep learning has shown great success in recognition and classification tasks in recent years. However, there is still a wide range of challenges when applying deep learning to the real world. Most deep neural networks and other machine learning models are trained under a static close-set scenario. However, the real world is more of an open-set scenario, in which it is difficult to collect samples that exhaust all classes. The problem of rejecting the unknown samples meanwhile accurately classifying the known classes is referred as Open Set Recognition (OSR) [4] or Open Category Learning [18].

In this chapter, we bring self-supervised pre-training to the OSR problem and fine-tune the pre-trained model with different types of loss functions: classification loss and representation loss. Particularly, we propose Detransformation Autoencoder (DTAE) for self-supervision. DTAE consists of three components: an encoder, a decoder, and an input transformation module. The encoder encodes all transformed images to representations, and the decoder reconstructs the representations back to the original images before transformations. Compared to the traditional autoencoder, DTAE learns the representations that describe the pixels and are invariant to the transformations. Our contribution in this chapter is threefold: First, we introduce DTAE as a self-supervised pre-training method for the OSR tasks. Second, our experiment results show that DTAE significantly improves the model performances for different down-streaming loss functions on several image datasets. Third, our analysis indicates that DTAE is able to capture some cluster information for both known and unknown samples even without class labels.

We organize the chapter as follows. Section 4.2 presents the self-supervision method, DTAE, in pre-training for the OSR tasks. Section 4.3 shows that the pre-training process can significantly improve the model performance in several standard image datasets. Meanwhile, the models pre-trained with DTAE achieve the best performance in detecting the unknown class and classifying the known classes.


Samples of known classes x
Input trans-formation module T
Transformed samples x_t
Encoder f
z
Decoder g
Reconstructed samples of known classes r_t
DTAE loss
Samples of known classes x
Encoder f
z
Classifier
Classification loss
Samples of known classes x
Encoder f
z
Representation loss

(a) Pre-training with DTAE

(b) Fine-tuning with classification loss

(c) Fine-tuning with representation loss

Figure 4.1: The two stage training process of the OSR problem.

4.2 Approach

We propose a two-step training process (pre-training step and fine-tuning step) for the OSR problems, thus better separating different classes in the feature space. As illustrated in Figure 4.1, the training process includes two steps: 1) pre-training step uses detransformation autoencoder (DTAE) to learn features for all the input data; 2) fine-tuning step uses representation loss functions or classification loss functions to learn discriminative features for different classes.

4.2.1 Pre-training step

The objective of the self-supervised pre-training process is to learn some meaningful representations via pretext tasks without semantic annotations. The desirable features should be invariant under input transformations, meanwhile, contain the essential information that can reconstruct the original input. We propose a detransformation autoencoder (DTAE) to learn representations by reconstructing ("detransforming") the original input from the transformed input. DTAE employs a transformation module and an encoder-decoder structure. While the encoder extracts the representations, the decoder reconstructs the original input from the learned representations.

The motivation of DTAE is to learn better representations for the OSR problem via encouraging intra-sample similarity and intra-class similarity of the learned representations. For example, if we have samples from "cat" class and "dog" class, given sample "cat1" and its transformation "cat1a", we can learn their representations z_{c1} and z_{c1a}. Similarly, we can learn the representations of "cat2" and "dog1" as z_{c2} and z_{d1}. The intra-sample similarity describes the similarity between the representations of the original input and its transformations, as z_{c1} and z_{c1a} in our example, and we denote this similarity as $sim(z_{c1}, z_{c1a})$. As the decoder in DTAE reconstructs the *same* original samples from the learned representations of *both* original and transformed samples, the learned representations are of high intra-sample similarity. Thus the learned representations are invariant to the transformations and contain important features of the samples. The intra-class similarity describes the similarity among the learned representations of the same class, as z_{c1} and z_{c2} in our example, and we denote this similarity as $sim(z_{c1}, z_{c2})$. The encoder-decoder structure in DTAE is a generative model that embeds crucial features in lower dimensions. Compared to a discriminate model, the representations learned by a generative model contain more comprehensive information to reconstruct the inputs. Thus, for a generative model, the learned representations of samples from the same class should be more similar than those of different classes. Overall, the desired representation space should satisfy $sim(z_{c1}, z_{c1a}) > sim(z_{c1}, z_{c2}) > sim(z_{c1}, z_{d1})$.

As shown in Figure 4.1a, in the pre-training stage with DTAE, the input transformation module T transforms any given data example x to several correlated views of the same example, denoted as $x_t = T(x)$. The network-based encoder $f(\cdot)$ extracts representation vectors from transformed data examples. Furthermore, decoder $g(\cdot)$ reconstructs the original data examples from the representation vectors. Let r_t denotes the reconstructed data example from transformed input x_t, then the detransformation loss function becomes:

$$\mathcal{L}(x, r_t) = \mathcal{L}(x, g(f(x_t))) \tag{4.1}$$

where $r_t = g(f(x_t))$. Specifically, we use MSE (Mean Squared Error) loss and have a total of M transformations, the loss function can be defined as:

$$\mathcal{L}_{\text{DTAE}} = \frac{1}{2} \sum_{t=0}^{M-1} \sum_{i=1}^{N} (x_i - r_{it})^2 \tag{4.2}$$

Each of the N data points has M transformations, and there are $M \times N$ data points after the input transformation module. In this work, we consider four transformations for each data example, i.e. $t \in \{0, 1, 2, 3\}$ for all N input examples, resulting in $4N$ data points. For this paper, the four transformations in our experiments are rotations of an image: 0, 90, 180, and 270 degrees.

4.2.2 Fine-tuning step

While the pre-trained network can be fine-tuned by different loss functions, we focus on two types of loss functions in this paper: the classification loss and the representation loss. 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 cross-entropy loss. 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 [80] and ii loss [41].

The fine-tuning network shares the same encoder and representation layer with the pre-training network. However, compared with the pre-training process, the fine-tuning process does not contain the input transformation module, which means the training examples are sent directly into the encoder. Moreover, instead of connecting to a decoder, the representation layer connects to a classification loss function or a representation loss function as shown in Figure 4.1b and Figure 4.1c. In this work, we consider both classification loss (cross-entropy loss) and representation loss (triplet loss [80] and ii loss [41]) in the OSR task.

4.2.3 Open Set Recognition (OSR)

A typical OSR task solves two problems: classifying the known classes and 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. Under this property, we propose the outlier score:

$$outlier\_score(x) = \min_{1 \leq i \leq C} \|\mu_i - z\|_2^2, \tag{4.3}$$

Where z is the learned representation of test sample x, μ_i is the representation centroid 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, the $P(y = i|x)$

is the output of the classification layer. Whereas in the case of a network without classification layer such as Figure 4.1c, we calculate $P(y = i|x)$ as:

$$P(y = i|x) = \frac{e^{-\|\mu_i - z\|_2^2}}{\sum_{j=1}^{C} e^{-\|\mu_j - z\|_2^2}} \tag{4.4}$$

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

$$y = \begin{cases} unknown, & \text{if } outlier\_score(x) > t \\ \operatorname*{argmax}_{1 \le i \le C} P(y = i|x), & \text{otherwise} \end{cases} \tag{4.5}$$

4.3 Experimental Evaluation

We evaluate the proposed pre-training method: Detransformation Autoencoder (DTAE) with simulated open-set datasets from the following datasets.
MNIST [58] contains 60,000 training and 10,000 testing handwritten digits from 0 to 9, which is 10 classes in total. Each example is a 28x28 grayscale image. 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.
Fashion-MNIST [93] is associated with 10 classes of clothing images. It contains 60,000 training and 10,000 testing examples. Same as the MNIST dataset, each example is a 28x28 grayscale image. 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 for testing.

CIFAR-10 [56] 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.

4.3.1 Evaluation Network Architectures and Evaluation Criteria

In the proposed method, we use self-supervision in the pre-training stage, and then in the second stage, we fine-tune the pre-trained model with two types of loss functions: classification loss and representation loss. Specifically, we use the cross-entropy loss as the example of classification loss, and use ii loss [41] and triplet loss [80] as the examples of representation loss. We first trained the model from scratch as a baseline (no pre-training) for each loss function and compared it with the corresponding fine-tuned models after self-supervision. Second, to evaluate our proposed self-supervision technique DTAE, we compare the model performance using DTAE with traditional Autoencoder (AE) and RotNet [27] in the pre-training stage. We also compare the proposed method with OpenMax [4] to show that it is effective to OSR problems.

Figure 4.1a illustrates the network architecture of the DTAE. Moreover, the hyper-parameters are different based on datasets. For the encoder of the MNIST and the Fashion-MNIST datasets, 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 layer. We use two fully connected non-linear layers with 256 and 128 hidden units after the convolutional component. Furthermore, the representation layer is six dimensions in our experiments. The representation layer is followed by a decoder, which is the reverse of the encoder in our experiments. We use the Relu activation function and set the Dropout rate as 0.2. We use Adam optimizer with a learning rate of 0.001. The encoder network architecture of the CIFAR-10 experiment is similar to the MNIST dataset, except the padded input layer is of size (36, 36). We use batch normalization in all the layers to prevent features from getting excessively large. And as mentioned in section 4.2.3, we use contamination ratio of 0.01 for the threshold selection. The encoder and representation layer maintain the same architecture and hyper-parameters in the fine-tuning network. Meanwhile, the decoder is replaced with different fully connected layers associated with different loss functions.

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. We chose the 10% FPR limit as higher FPR is generally undesirable, particularly when negative instances are much more abundant than positive instances. We measure the F1 scores for known and unknown classes separately as one of the OSR tasks is to classify the known classes. Moreover, we perform t-tests with 95% confidence in the AUC scores and F1 scores to see if the proposed DTAE pre-training method can significantly improve different loss functions.

Table 4.1: The average ROC AUC scores of 30 runs at 100% and 10% FPR of OpenMax and a group of 5 methods (without pre-training as baseline, pre-training with AE, RotNet, DTAE and TAE) for each of the 3 loss functions (ce, ii, triplet). The underlined values are statistically significantly better than the baselines via t-test with 95% confidence. The values in bold and in brackets are the highest and the second-highest values in each group.

		MNIST		Fashion-MNIST		CIFAR-10	
FPR		100%	10%	100%	10%	100%	10%
OpenMax		0.9138	0.0590	0.7405	0.0160	0.6750	0.0060
ce	No pre-training	0.9255	0.0765	0.7175	0.0300	0.5803	0.0070
	AE	0.9410	**0.0805**	0.7346	0.0300	$\underline{0.6114}$	[0.0084]
	RotNet	0.9367	0.0769	0.7364	[0.0316]	[0.6124]	$\underline{0.0083}$
	DTAE (ours)	$\underline{\textbf{0.9523}}$	[0.0801]	$\underline{\textbf{0.7490}}$	**0.0324**	$\underline{\textbf{0.6183}}$	$\underline{\textbf{0.0086}}$
	TAE	[$\underline{0.9477}$]	0.0799	[$\underline{0.7389}$]	0.0298	$\underline{0.6012}$	0.0075
ii	No pre-training	**0.9578**	0.0821	0.7684	0.0399	0.6392	0.0103
	AE	0.9560	**0.0828**	0.7636	0.0377	0.6320	0.0098
	RotNet	0.9530	0.0813	[0.7703]	[0.0404]	[0.6478]	[0.0106]
	DTAE (ours)	[0.9566]	[0.0825]	**0.7802**	**0.0410**	$\underline{\textbf{0.6520}}$	**0.0108**
	TAE	0.9515	0.0815	0.7657	0.0387	0.6214	0.0091
triplet	No pre-training	0.9496	0.0750	0.7160	0.0211	0.6106	0.0089
	AE	**0.9563**	**0.0772**	0.7254	0.0220	0.6251	0.0090
	RotNet	0.9342	0.0702	[$\underline{0.7435}$]	$\underline{\textbf{0.0252}}$	[$\underline{0.6285}$]	[0.0095]
	DTAE (ours)	[0.9543]	[$\underline{0.0758}$]	$\underline{\textbf{0.7441}}$	[$\underline{0.0234}$]	$\underline{\textbf{0.6327}}$	**0.0096**
	TAE	0.9531	0.0757	0.7271	0.0215	0.6114	0.0081

4.3.2 Experimental Results

Model performance We compare the model performances of cross-entropy loss, ii loss, and triplet loss with and without pre-training. Table 4.1 are the averaged ROC AUC scores of the model performances in three datasets under different FPR values. Comparing "RotNet", "DTAE" and "AE" rows with "No pre-training" rows, we observe that using self-supervision techniques for pre-training significantly improved the model performance. The results also show that our proposed self-supervision method DTAE achieves the top two ROC AUC scores for all the cases. Moreover, with our proposed pre-training method, all three loss functions perform better than OpenMax in 5 out of 6

Table 4.2: The average F1 scores of 30 runs of OpenMax and a group of 5 methods (without pre-training as baseline, pre-training with AE, RotNet, DTAE and TAE) for each of the 3 loss functions (ce, ii, triplet). The underlined values show statistically significant improvements (t-test with 95% confidence) comparing to the baselines. The values in bold and in brackets are the highest and the second highest values in each group.

		MNIST			Fashion-MNIST			CIFAR-10		
		Known	Unknown	Overall	Known	Unknown	Overall	Known	Unknown	Overall
OpenMax		0.8964	0.7910	0.8814	0.7473	0.5211	0.7150	0.6456	0.5407	0.6307
ce	No pre-training	0.7596	0.7561	0.7591	0.6858	0.5591	0.6677	0.5672	0.3697	0.5390
	AE	0.7735	0.7894	0.7757	0.7264	0.5481	0.7009	0.5729	0.4605	[0.5569]
	RotNet	[0.8931]	[0.8447]	[0.8862]	0.7117	**0.5694**	0.6914	0.5616	**0.4729**	0.5489
	DTAE (ours)	**0.8967**	**0.8579**	**0.8912**	[0.7335]	[0.5692]	[0.7100]	**0.5911**	[0.4728]	**0.5742**
	TAE	0.8804	0.8420	0.8749	**0.7482**	0.5364	**0.7179**	[0.5815]	0.3889	0.5540
ii	No pre-training	0.9320	0.8833	0.9250	0.7720	0.5870	0.7456	0.6206	0.3570	0.5829
	AE	**0.9387**	**0.8950**	**0.9325**	0.7669	0.5745	0.7394	0.6241	0.2527	0.5711
	RotNet	0.9300	0.8761	0.9223	**0.7771**	**0.6108**	**0.7533**	**0.6442**	[0.3980]	[0.6090]
	DTAE (ours)	[0.9344]	[0.8885]	[0.9279]	[0.7768]	[0.6064]	[0.7524]	[0.6421]	**0.4252**	**0.6111**
	TAE	0.9308	0.8830	0.9240	0.7625	0.5869	0.7374	0.6135	0.2103	0.5559
triplet	No pre-training	0.9103	0.8302	0.8989	0.7491	0.5055	[0.7208]	0.5798	0.4515	0.5614
	AE	[0.9144]	0.8356	[0.9032]	0.7505	0.5051	0.7154	[0.6086]	[0.4800]	0.5902
	RotNet	0.9012	0.8182	0.8893	[0.7514]	[0.5376]	[0.7208]	0.6037	**0.4978**	[0.5886]
	DTAE (ours)	**0.9166**	**0.8513**	**0.9073**	**0.7558**	**0.5459**	**0.7259**	**0.6205**	0.4724	**0.5993**
	TAE	0.9126	[0.8387]	0.9021	0.7472	0.5092	0.7132	0.5926	0.4220	0.5682

cases (3 datasets×2 FPR limits).

To evaluate the detransformation component of DTAE, we performed an ablation study on our method without detransformation, which is denoted as TAE. Although both DTAE and TAE use transformed instances as input, TAE reconstructs the transformed instances as output, while DTAE reconstructs the original instances as output. Comparing the "TAE" rows and "DTAE" rows, we observe that the detransformation component in DTAE plays a key role in improving the model performance. That is, our results indicate that learning features invariant to transformations, via detransformation, can yield more effective features than those learned from reconstructing the same samples.

Table 4.2 shows that the OSR performances of different methods are measured by F1 scores in known and unknown class domains. We first calculate

Table 4.3: The training time (in seconds) for the self-supervision methods in different datasets.

	AE	RotNet	DTAE
MNIST	75	132	137
Fashion-MNIST	70	118	145
CIFAR-10	86	147	182

the F1 scores for each known class and the unknown class, then average all the classes as the Overall F1 scores. The results show that models with pre-training achieve statistically significant improvements. Moreover, Our proposed method also achieves the top two F1 scores in 26 out of 27 cases (3 loss functions$\times$3 datasets$\times$3 domains).

Training time While the pre-training step benefits the model performances and does not affect the final model complexity and inference time, it takes extra time during the training phase. Table 3.3 shows the comparison of the training time of the self-supervised networks in different datasets via NVIDIA RTX 2080. Because RotNet and DTAE both include transformed data as input, they took a longer training time than AE. We observe that DTAE takes a slightly longer training time than RotNet. The reason is that the network structure of DTAE is more complex than that of RotNet. While both RotNet and DTAE share the same encoder and representation layer structures, RotNet uses a softmax layer after the representation layer. Meanwhile, DTAE connects the representation layer with a decoder module. The decoder is the reverse of the encoder, which contains more layers than a softmax layer and needs a longer time in the forward and backward propagations.

Openness study We also study the model performances against vary Openness [78]. Let n_{train} be the number of known classes participant in the training

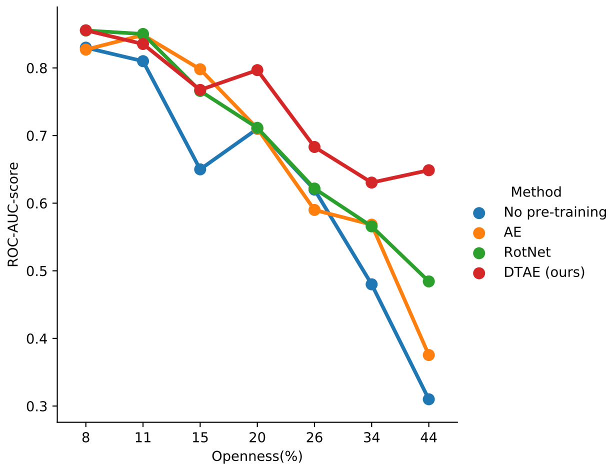


Figure 4.2: AUC-ROC scores against varying Openness.

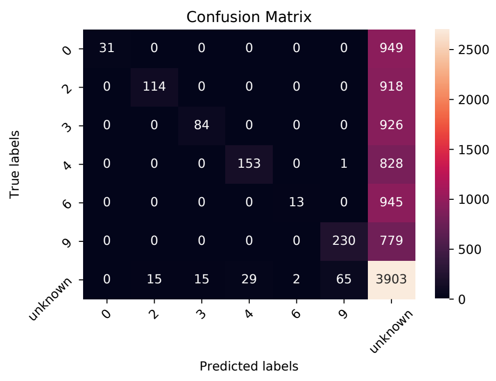
phase, with n_{test} denotes the number of classes in the test set, and n_{target} denotes the number of classes to be recognized in the testing phase. Openness can be defined as: $Openness = 1 - \sqrt{\frac{2 \times n_{train}}{n_{test}+n_{target}}}$.

In our experiments with the Fashion-MNIST dataset, we use all the ten classes in testing phase ($n_{test} = 10$) and varying the number of known classes from 2 to 9 ($n_{train} = 2, \ldots, 9$) in the training phase, and remaining classes together are treated as the unknown class to be recognized along with the known classes during inference ($n_{target} = n_{train} + 1$). That is, the openness is varied from 8% to 44%. We evaluate the AUC ROC scores of different models using cross-entropy loss: without pre-training (baseline), pre-training with AE, pre-training with RotNet, and pre-training with our proposed DTAE. The results are shown in Figure 4.2. We observe that the three different models have similar performances when the openness is small. However, the AUC ROC scores of the baseline (No pre-training) degrade rapidly as the Openness increases. Moreover, the trend is alleviated by pre-training with self-supervision methods. Overall, the model pre-trained with DTAE is relatively more robust to openness and achieves the best performance.

4.3.3 Analysis

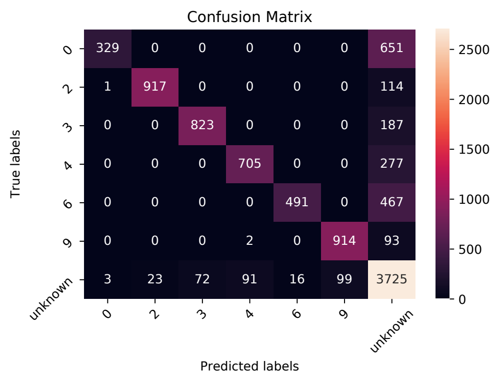
In this section, we empirically analyze some properties of the proposed method related to the learned representations. Figure 4.3 shows the confusion matrices of the experiments with the MNIST test set. In these experiments, digits “0”, “2”, “3”, “4”, “6” and “9” are known classes while the remaining classes are unknown and absent from training set. Figure 4.3a shows the confusion matrix of the model using cross-entropy loss without pre-training. Figures 4.3b and 4.3c show the confusion matrices of the learned representations of the model pre-trained by RotNet and DTAE and fine-tuned by cross-entropy, respectively. According to the true positive predictions along the diagonals of the matrices. Both pre-training methods benefit the model performance. Moreover, we observe that RotNet has relatively poor performance on digit “0” and DTAE almost doubles the true positive predictions on digit “0” compared to RotNet. As the objective of RotNet is to predict the image rotations. However, the features learned in this method cannot benefit rotation agnostic images [20], such as round objects like digit “0”. The objective of DTAE, on the other hand, is to learn the representations that are invariant to rotations/transformations. Thus the ambiguity of the orientations does not affect the model performance.

To analyze the differences in representations after pre-training and after fine-turning, we plot 1000 samples from the Fashion-MNIST test set in Figure 4.4. In these experiments, classes “T-shirt/top”, “Pullover”, “Dress”, “Coat”, “Shirt” and “Ankle boot” are known classes while the remaining classes are unknown and absent from the training set. Figure 4.4a shows



	0	2	3	4	6	9	unknown
0	31	0	0	0	0	0	949
2	0	114	0	0	0	0	918
3	0	0	84	0	0	0	926
4	0	0	0	153	0	1	828
6	0	0	0	0	13	0	945
9	0	0	0	0	0	230	779
unknown	0	15	15	29	2	65	3903

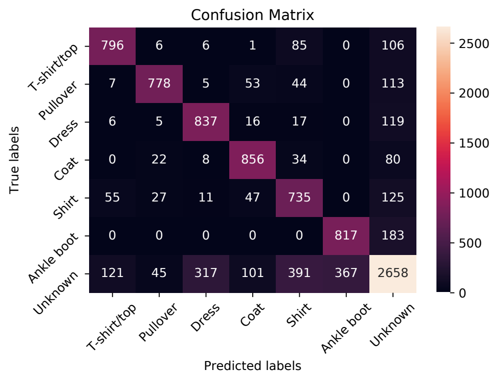
(a) Without pre-training (CE)



	0	2	3	4	6	9	unknown
0	329	0	0	0	0	0	651
2	1	917	0	0	0	0	114
3	0	0	823	0	0	0	187
4	0	0	0	705	0	0	277
6	0	0	0	0	491	0	467
9	0	0	0	2	0	914	93
unknown	3	23	72	91	16	99	3725

b

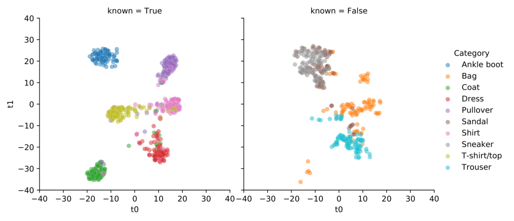
(b) After fine-tuning (RotNet + CE)



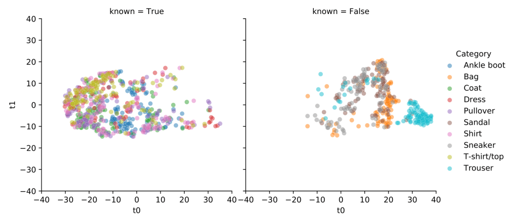
	T-shirt/top	Pullover	Dress	Coat	Shirt	Ankle boot	Unknown
T-shirt/top	796	6	6	1	85	0	106
Pullover	7	778	5	53	44	0	113
Dress	6	5	837	16	17	0	119
Coat	0	22	8	856	34	0	80
Shirt	55	27	11	47	735	0	125
Ankle boot	0	0	0	0	0	817	183
Unknown	121	45	317	101	391	367	2658

(c) After fine-tuning (DTAE + CE)

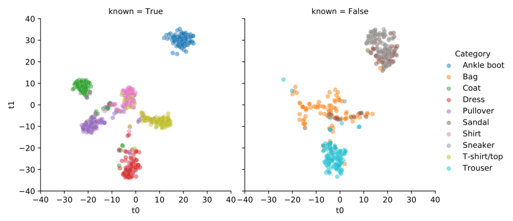
Figure 4.3: The confusion matirces of the MNIST dataset in different experiments using cross-entropy loss: (a) training without pre-training (CE); (b): pre-training with RotNet (RotNet + CE); (c): pre-training with DTAE (DTAE + CE).



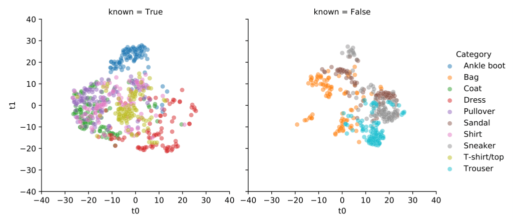
(a) Without pre-training (CE)



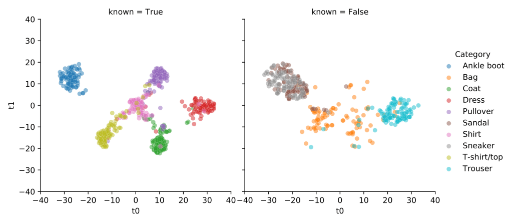
(b) After pre-training (RotNet)



(c) After fine-tuning (RotNet + CE)



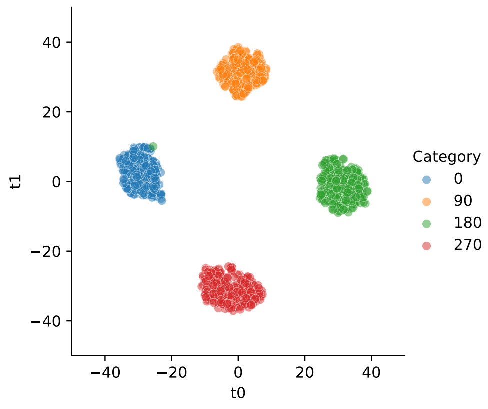
(d) After pre-training (DTAE)



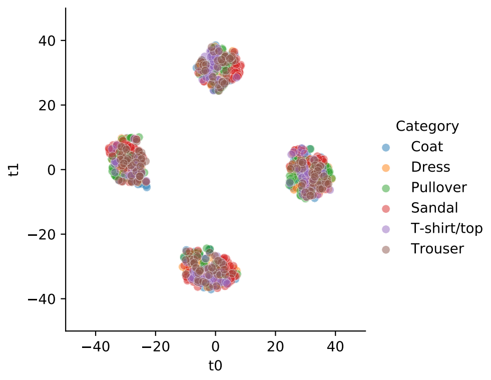
(e) After fine-tuning (DTAE + CE)

Figure 4.4: The t-SNE plots of the Fashion-MNIST test set using cross-entropy loss. The left subplots are the representations of the known class, and the right plots are the representations of the unknown classes.

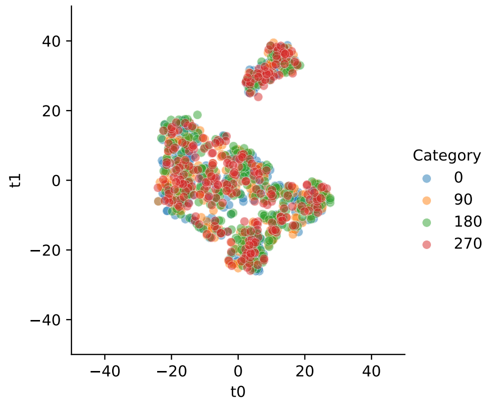
the t-SNE plot of the representations learned from cross-entropy loss without pre-training. Figures 4.4b and 4.4c are the learned representations of the model pre-trained by RotNet and fine-tuned by cross-entropy in different stages. Figures 4.4d and 4.4e are the learned representations of the model pre-trained by DTAE and again, fine-tuned by cross-entropy. From all the final representations of the three models in Figures 4.4a, 4.4c and 4.4e, we observe overlaps between the known class "Ankle boot" (blue) and one component of the unknown class "Sneaker" (gray) as well as class "Dress" (red) and class "Trouser" (cyan). And the pre-training reduces the overlaps between "Shirt" (pink) and "Bag" (orange). Moreover, for the representations



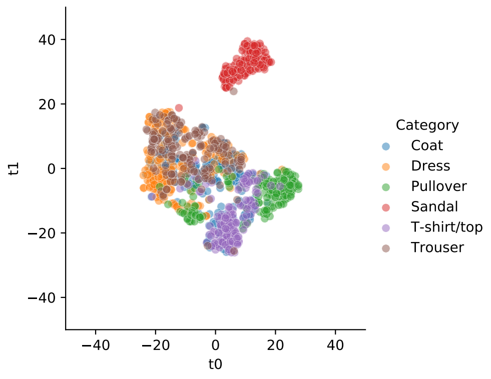
(a) Trans. (RotNet)



(b) Fash. (RotNet)



(c) Trans. (DTAE)

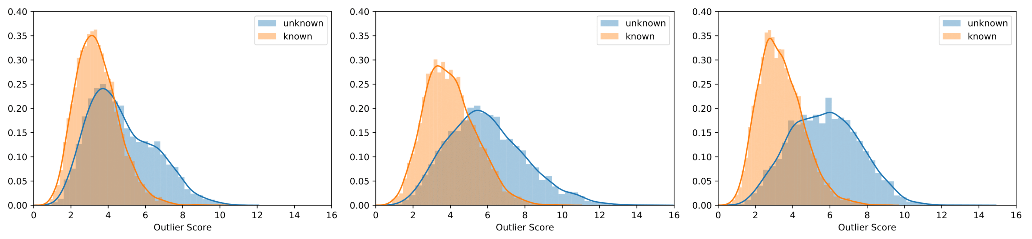


(d) Fash. (DTAE)

Figure 4.5: The t-SNE plots of the learned representations for the Fashion-MNIST training set in the pre-training stage: (a): the representations on transformation (Trans.) classes learned by RotNet; (b): the representations on fashion (Fash.) classes learned by RotNet; similarly for DTAE in (c) and (d).

after pre-training, it shows that the representations learned by DTAE in Figure 4.4b are more separable than those learned by RotNet in Figure 4.4d for different classes. Note that DTAE, similar to RotNet, is not provided with class labels, but it can find representations that are more separable among the classes than RotNet.

To understand what kind of information is in the learned representations after pre-training, we analyze how the representations are associated with the transformation classes and fashion (target) classes. We color the representations of the Fashion-MNIST training set based on transformation and fashion classes separately in Figure 4.5. If the representations have much information for the transformation classes, representations in the same transformation class (color) will be clustered together; otherwise, they will be dispersed. Similarly, we can use the same technique to understand the amount of fashion information in the learned representations. By comparing Figures 4.5a and 4.5b, we observe that RotNet learns more transformation than fashion information in the representations. However, from Figures 4.5c and 4.5d, we find that DTAE learns more fashion than transformation information in the representations. Therefore, RotNet is more biased towards the transformation information, while DTAE is more biased towards the fashion information, which is more desirable in our case. Moreover, from Figures 4.5a and 4.5c, we observe that the representations learned by DTAE contain much less transformation information than those learned by RotNet. However, from Figures 4.5b and 4.5d, we find that the representations learned by DTAE contain more fashion (target) information than those learned by RotNet. In summary, DTAE is not only more biased towards fashion (target) than transformation information, and it also learns more fashion information



(a) Without pre-training (b) After fine-tuning (RotNet) (c) After fine-tuning (DTAE)

Figure 4.6: The distributions of outlier scores for the known and unknown classes of the Fashion-MNIST dataset in different experiments using cross-entropy loss.

and less transformation information than RotNet.

Figure 4.6 shows distributions of the outlier scores in experiments on the Fashion-MNIST test set. Compared with the model without pre-training in Figure 4.6a, the pre-training steps in Figures 4.6b and Figure 4.6c increase the outlier scores in the unknown classes, which pushes their score distributions further away from the known classes. The fact that there are fewer overlaps between the known classes and the unknown class makes them more separable. The results indicate that the model pre-trained with DTAE has the fewest overlaps between the known and unknown classes.

4.4 Conclusion

In this chapter, we introduce the self-supervision technique to OSR problems. We provide experiments across different image datasets to measure the benefits of the pre-training step for OSR problems. Moreover, we have presented a novel method: Detransformation Autoencoder (DTAE) for self-supervision. The proposed method engages in learning the representations that are invariant to the transformations of the input data. We evaluate the pre-trained model with both classification and representation loss functions. The experiments on several standard image datasets show that the proposed method significantly outperforms the baseline methods and other self-supervision techniques. Our analysis indicates that DTAE outperforms traditional AE and also yields representations that contain more target class information and less transformation information than RotNet.

Chapter 5

Representation Learning with Function Call Graph Transformations for Malware Open Set Recognition

5.1 Introduction

As machine learning has achieved great success in various domains, there is still a wide range of challenges in the real world. For example, from the security scenario, new malware classes emerge daily. A robust machine learning system for malware detection should be able to classify the known malware classes and recognize the newly unknown malware classes, which is referred as Open Set Recognition (OSR) problem [4]. The OSR problem aims to classify the multiple known classes for a multinomial classification problem while identifying the unknown classes.

In this chapter, we follow a two-stage learning approach to address the OSR problem in malware classification. Given the malware assembly files, we first extract the function call graphs (FCGs) as the input representations of the malware samples. Next, to learn better representations for the malware samples, we use a self-supervised pre-training approach for the extracted FCGs. As the self-supervised learning approach needs a pretext task, we propose two transformations for the FCG inputs. Then both original and transformed FCGs are processed by a detransformation autoencoder (DTAE) [48]. DTAE involves an encoder and a decoder. The encoder learns the representations for the inputs while the decoder reconstructs the transformed inputs back to their original forms. After pre-training and fine-tuning the representations, we apply a statistical thresholding approach to find the optimal threshold for the OSR tasks. Our contributions include, first, we summarize the characteristics of the malware FCGs. Second, we propose two transformation methods for the malware FCGs to facilitate the self-supervised pre-training process for the OSR tasks. Third, we introduce a statistical thresholding approach for the OSR task, which performs similarly to the manually selected threshold. Finally, our experiments on two different malware datasets indicate that our proposed self-supervised pre-training approach improves the model performance on the OSR tasks.

We organize this chapter as follows. In section 5.2, we first present our proposed approach to the self-supervised pre-training for the malware FCGs, then introduce a statistical thresholding approach for the OSR tasks. Finally, section 5.3 evaluates the proposed approach through experiment setup and results from the analysis.

5.2 Approach

The objective of open set recognition (OSR) is to classify the known classes and the unknown classes even when the collected training samples cannot exhaust all the classes. An advanced malware classification system that utilizes OSR techniques can classify the known malware families while identifying the unknown malware family. Hassen and Chan [39] convert malware assembly files to FCGs as OSR input. Here, we also use the FCGs as input samples. To learn better representations for the OSR problem in malware classification, we introduce a self-supervised pre-training process to learn low-level features of the malware samples. Based on the FCGs characteristics, we propose two transformation methods for malware FCGs to facilitate the pretext task. Moreover, we introduce a statistical method to identify unknown instances.

5.2.1 Malware Function Call Graphs (FCGs)

Previous research works have proposed various ways to extract features for malware classifications: Schultz et al. [82] extract features from printable strings in malware binaries. Hu et al. [45] extract features from instruction n-grams. Hassen and Chan [39] convert malware assembly files to FCGs as input features. The FCGs can better preserve structural information between functions. Thus, in this paper, we adopt the same FCGs as in [39]. The system first extracts FCG representations from dissembled binaries. In the FCGs, the vertices are functions, and edges are the interactions (calls) between functions. Then based on the instruction opcode sequence, it clusters the functions using Locality Sensitive Hashing (LSH), and the vertices (functions) are then arbitrarily labeled with cluster-ids.

Table 5.1: Graph statistics for datasets in function call graphs (FCGs), biochemical molecules (BMs) and social networks (SN). The statistics includes: average number of vertices, average number of degrees and % of vertices that are neighbors (Degree/Vertex), average number of connected components (C.C.), average size of each connected components and relative connected components size (C.C. Size/Vertex).

Dataset	Category	Vertex	Degree (/Vertex)	C.C.	C.C. Size (/Vertex)
MS	FCGs	27.55	1.66(6%)	14.99	3.74(16%)
AG	FCGs	31.73	3.31(10%)	16.97	2.28(7%)
MUTAG	BMs	17.93	1.10 (6%)	3.49	5.86(33%)
PROTEINS	BMs	39.06	1.86(5%)	4.75	9.78(25%)
COLLAB	SNs	74.49	32.99(44%)	4.65	30.36(41%)
DBLP_v1	SNs	10.48	1.87(18%)	1.93	6.12(58%)

The extracted FCGs are directed graph representations of the dissembled malware binaries, with function clusters as the graph vertices and the caller-callee relations between functions as graph edges. As the cluster ids are arbitrarily assigned, we will get different isomorphic graphs for the same malware binaries when we change the order of the cluster ids.

5.2.2 FCG characteristics

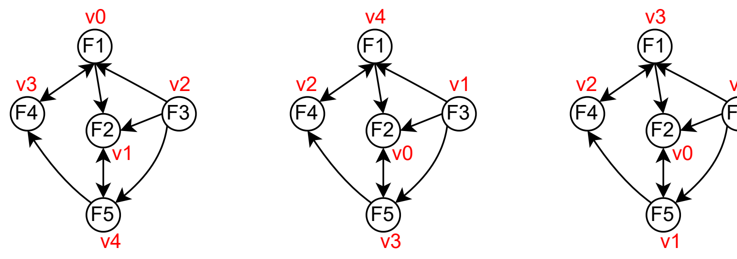
In this subsection, we compare the characteristics of the FCGs of malware datasets with two other categories of graphs: biomedical molecules (BMs) and social networks (SNs) in Table 5.1. Specifically, we compare the FCGs extracted from two malware datasets: Microsoft Challenge (MC) and Android Genome (AG) (see section 5.3 for more details) with MUTAG [55], PROTEINS [7], COLLAB [97] and DBLP_v1 [30]. In the table, “Vertex” and “Degree” are the average numbers of vertices and degrees in each dataset. We also measure the average percentage of vertices that are neighbors by dividing the number of degrees by the number of vertices. Moreover, we calculate the average number of connected components (C.C.) and the average size of connected components (C.C. Size) for each dataset. Also, we divide the size of the C.C. by the number of vertices to measure the relative C.C. Size. Comparing the graph statistics of the FCGs with the other categories, we conclude two characteristics of the FCGs.

First, FCGs are sparser (i.e., have fewer direct neighbors) than the graphs from the other two categories, especially social networks. In the COLLAB dataset, the average degree of a graph is 32.99, which means 44% of the vertices are direct neighbors. Meanwhile, 6% of vertices are direct neighbors in the MS dataset and 10% for the AG dataset.

Second, FCGs have more and relatively small connected components than the other two categories of graphs. From Table 5.1, both malware FCGs contain around 15 connective components, while the datasets from the other two categories contain less than five connected components. Furthermore, the average sizes of each connected component in the two malware FCGs are less than 4, which means less than four vertices are connected while isolated from the rest of the vertices. Especially for the AG dataset, the connected components are of size 2.28 on average, which is only 7% of the total vertices. The relative connected components size is above 25% of the total vertices for the four datasets from the other two categories. Notably, the relative size of connected components in DBLP_v1 dataset reaches 58%.

5.2.3 FCG transformations

Self-supervised learning generally involves input transformations to achieve pretext tasks to learn better representations of input samples. The research in [100] finds that the optimal input transformation method is task-relevant, and it concludes that node dropping and subgraph sampling are generally



	F1	F2	F3	F4	F5
F1	0	1	0	1	0
F2	0	0	0	0	1
F3	1	1	0	0	1
F4	1	0	0	0	0
F5	0	1	0	1	0

(a) Original

	F2	F3	F4	F5	F1
F2	0	0	0	1	0
F3	1	0	0	1	1
F4	0	0	0	0	1
F5	1	0	1	0	0
F1	1	0	1	0	0

(b) FCG-shift

	F2	F5	F4	F1	F3
F2	0	1	0	0	0
F5	1	0	1	0	0
F4	1	0	0	1	0
F1	1	0	1	0	0
F3	1	1	0	1	0

(c) FCG-random

Figure 5.1: Transformations of FCG adjacency matrix

beneficial across biochemical molecules and social networks datasets. The node dropping transformation creates a new graph view by discarding a specific set of vertices and edges from the original input graph. As the FCGs have fewer direct neighbors and are sparser than other graph datasets, discarding vertices and their edges will remove more neighborhood information. Thus the node dropping transformation is less applicable to the malware FCGs. The subgraph sampling transformation creates a new graph view by sampling a subgraph from the original input graph via a random walk. From the second characteristic of the FCGs, the FCGs contain more connected components (around 15 for the FCGs dataset from Table 5.1). Since a random walk subgraph sampling will keep one connected component and discard the rest (14 out of 15), the subgraph sampling will discard more than 90% information. Thus subgraph sampling is not desirable in learning the representations of the FCGs.

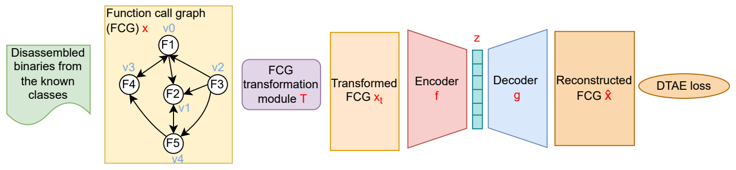
As FCGs can be represented by adjacency matrices, and the ordering of vertices in the matrices is arbitrary. Here, we propose two types of transformations: FCG-shift and FCG-random for the malware FCGs. The two transformations generate a new isomorphic view by altering the ordering of vertices. Given the original order of clusters-ids assignment as Figure 5.1a, the FCG-shift transformation randomly select a pivots n, and then shift the cluster-ids assignments n positions to the left. For example, in Figure 5.1b, the order of vertices (cluster-ids) is shifted one position to the left. The original vertex order "F1", "F2", "F3", "F4", "F5" becomes "F2", "F3", "F4", "F5", "F1". The FCG-random transformation randomly permute the order of vertices and generated new adjacency matrices based on the permuted vertex order. In Figure 5.1c, after the random permutation, the original vertex order "F1", "F2", "F3", "F4", "F5" becomes "F2", "F5", "F4", "F1", "F3". Both FCG-shift and FCG-random maintain the orignal FCGs' properties without information loss by generating isomorphic graphs to the original graphs.

5.2.4 Representation Learning

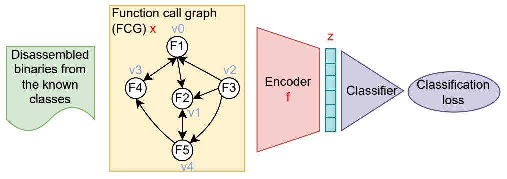
In this work, we follow the two-stage learning strategy to learn the representations of input malware FCGs. We adopt the self-supervised learning strategy to initial the network with low-level representations in the first stage. In the second stage, we fine-tune the network with different loss functions to extract the discriminative representations.

5.2.4.1 Pre-training stage

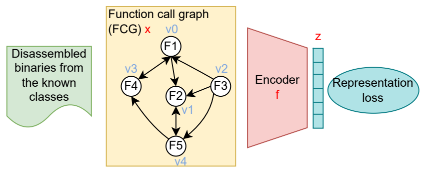
With the proposed FCG transformations, we adopt detransformation autoencoder (DTAE) proposed in [48] as our pretext task here to pre-train the the


Disassembled binaries from the known classes
Function call graph (FCG) x
v0
F1
v3
F4
F2
F3
v2
v1
F5
v4
FCG transformation module T
Transformed FCG x_t
Encoder f
z
Decoder g
Reconstructed FCG $\hat{x}$
DTAE loss

(a) Pre-training step


Disassembled binaries from the known classes
Function call graph (FCG) x
v0
F1
v3
F4
F2
F3
v2
v1
F5
v4
Encoder f
z
Classifier
Classification loss

(b) Fine-tuning step with classification loss


Disassembled binaries from the known classes
Function call graph (FCG) x
v0
F1
v3
F4
F2
F3
v2
v1
F5
v4
Encoder f
z
Representation loss

(c) Fine-tuning step with representation loss

Figure 5.2: The training process of using detransformation autoencoder.

network. As depict in Figure 5.2a, given an input disassembled binaries of the malware samples from the known classes, we first extract its FCG x_i. Then the FCG transformation module $T(.)$ transforms the original FCG to its correlated views x_{it}. Next, the encoder $f(.)$ learns the representations z of the transformed FCG x_{it}, and the decoder reconstruct the representation z back to its original FCG format $\hat{x}_{it}$. Assuming we have M transformations for N FCG inputs. The learning process of neural network-based encoder-decoder structure is guided by DTAE loss:

$$\mathcal{L}_{\text{DTAE}} = \frac{1}{2} \sum_{t=0}^{M-1} \sum_{i=1}^{N} (x_i - \hat{x}_{it})^2 \tag{5.1}$$

In this paper, we transform FCGs four times for each experiment, i.e., $M = 4, t \in \{0, 1, 2, 3\}$.

5.2.4.2 Fine-tuning stage

After pre-training the neural network with transformed inputs, we fine-tune the encoder and presentation layer (z) with the original inputs for the downstream tasks. Here, we consider two types of loss functions for fine-tuning: classification loss and representation loss. The objective of classification loss is to explicitly lower the training data's classification error in the decision layers, such as cross-entropy loss. When using classification loss as the fine-tuning loss function, we connect the presentation layer with a classifier, which associates with a classification loss function as shown in Figure 5.2b. 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. When using representation loss as the fine-tuning loss function, we directly constrain the representation layer with a representation loss function, as shown in Figure 5.2c.

5.2.5 Open Set Recognition

After fine-tuning the encoder with the original FCG inputs, we extract the learned representations z for the malware input. We utilize the distances between the representations for the open set recognition (OSR) task: classifying the known classes and identifying the unknown class.

For a known class k that participant in the training process, we first find its representation centroid as prototype μ_k. Given the representation z_i for sample i from class k (i.e. $y_i = k$), we can calculate the prototype as:

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{N_k} z_i, \tag{5.2}$$

where N_k is the number of samples in class k. After obtaining the prototypes, we introduce a statistical method to perform the OSR task. Specifically, we calculate the mean m_k and standard deviation s_k of the distances

d_i from the training samples to the prototype k.

$$m_k = \frac{1}{N_k} \sum_{i=1}^{N_k} d_i \tag{5.3}$$

$$s_k = \sqrt{\frac{\sum_{i=1}^{N_k} (z_i - m_k)^2}{N_k}} \tag{5.4}$$

Then we normalize the distances between representations and prototypes based on the prototypes' means and standard deviations, and calculate the outlier score based on the least of standard deviations to the prototype :

$$outlier\_score(x) = \min_{1 \le k \le C} \frac{\|D(\mu_k, z) - m_k\|}{s_k}, \tag{5.5}$$

where C is the number of known classes, and z is the learned representation of input x. $D(.,.)$ is a distance function, and we use euclidean distances in this paper. Based on the Empirical Rule, a test instance can be recognized as "unknown" if its outlier score is more significant than three standard deviations.

$$y = \begin{cases} unknown, & \text{if } outlier\_score(x) > 3 \\ \operatorname*{argmin}_{1 \le k \le C} \frac{\|D(\mu_k, z) - m_k\|}{s_k}, & \text{otherwise} \end{cases} \tag{5.6}$$

5.3 Experiments

We evaluate the proposed self-supervised pre-training method with two types of downstream loss functions: triplet loss [80] (representation loss) and cross-entropy loss (classification loss). Moreover, we test the proposed approach on two malware datasets:

Table 5.2: The average AUC scores of 30 runs at 100% and 10% FPR of OpenMax and a group of 5 methods for each of the two types of loss functions (ce and triplet): without pre-training, pre-training via DTAE with transformations node dropping (ND), Subgraph sampling (SS), FCG-shift and FCG-random. The values in bold are the highest values in each group. The underlined values show statistically significant improvements (t-test with 95% confidence) comparing with OpenMax.

		OpenMax	ce	triplet
	FPR		No pre-training / ND / SS / FCG-shift (ours) / FCG-random (ours)	No pre-training / ND / SS / FCG-shift (ours) / FCG-random (ours)
MC	100%	$0.880_{\pm 0.037}$	$0.918_{\pm 0.036}$ / $0.914_{\pm 0.063}$ / $0.626_{\pm 0.054}$ / $\underline{0.938_{\pm 0.015}}$ / $\underline{\mathbf{0.947}_{\pm \mathbf{0.011}}}$	$0.929_{\pm 0.020}$ / $0.919_{\pm 0.032}$ / $0.723_{\pm 0.071}$ / $\underline{0.932_{\pm 0.017}}$ / $\underline{\mathbf{0.933}_{\pm \mathbf{0.015}}}$
	10%	$0.040_{\pm 0.003}$	$0.053_{\pm 0.008}$ / $0.053_{\pm 0.014}$ / $0.018_{\pm 0.005}$ / $\underline{0.061_{\pm 0.003}}$ / $\underline{\mathbf{0.063}_{\pm \mathbf{0.003}}}$	$0.058_{\pm 0.004}$ / $0.056_{\pm 0.006}$ / $0.036_{\pm 0.008}$ / $\underline{\mathbf{0.061}_{\pm \mathbf{0.003}}}$ / $\underline{\mathbf{0.061}_{\pm \mathbf{0.003}}}$
AG	100%	$0.457_{\pm 0.200}$	$0.852_{\pm 0.056}$ / $0.820_{\pm 0.128}$ / $0.418_{\pm 0.080}$ / $\underline{\mathbf{0.865}_{\pm \mathbf{0.060}}}$ / $\underline{0.854_{\pm 0.062}}$	$0.868_{\pm 0.046}$ / $0.818_{\pm 0.124}$ / $0.427_{\pm 0.094}$ / $\underline{0.873_{\pm 0.036}}$ / $\underline{\mathbf{0.883}_{\pm \mathbf{0.035}}}$
	10%	$0.001_{\pm 0.001}$	$0.021_{\pm 0.012}$ / $0.019_{\pm 0.016}$ / $0.002_{\pm 0.002}$ / $\underline{\mathbf{0.022}_{\pm \mathbf{0.013}}}$ / $\underline{0.019_{\pm 0.009}}$	$0.024_{\pm 0.010}$ / $0.018_{\pm 0.011}$ / $0.002_{\pm 0.002}$ / $\underline{0.025_{\pm 0.011}}$ / $\underline{\mathbf{0.027}_{\pm \mathbf{0.011}}}$

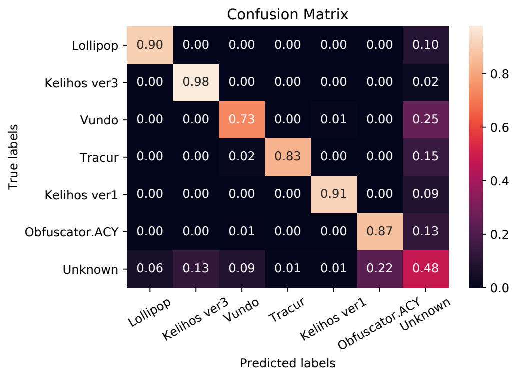
Microsoft Challenge (MC) [74] contains disassembled malware samples from 9 families:"Ramnit", "Lollipop", "Kelihos ver3", "Vundo", "Simda", "Tracur", "Kelihos ver1", "Obfuscator.ACY " and "Gatak". We use 10260 samples that can be correctly parsed then extracted their FCGs as in [39] for the experiment. 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 that only exist in the test set.

Android Genome (AG) consists of 1,113 benign android apps and 1,200 malicious android apps. The benign samples are provided by our colleague, and the malicious samples are from [108]. We select nine families with a relatively larger size for the experiment to be fairly split into the training and test sets. The nine families contain 986 samples in total. We first use [23] to extract the function instructions and then generated the FCGs as in [39]. Also, to simulate an open-world scenario as the MC dataset, we randomly pick six digits as the known classes in the training set while considering the rest as the unknown class, which are only used in the test phase.

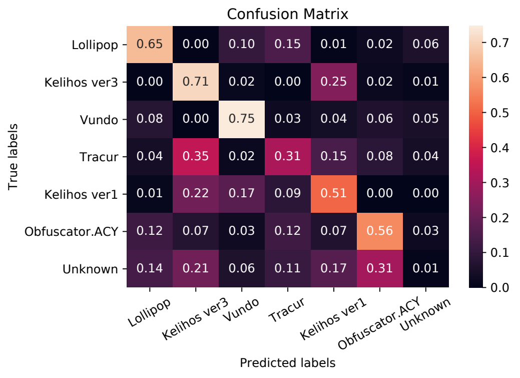
5.3.1 Experimental Setup

As described in Section 5.2, our proposed approach first extracts the FCGs from the malware samples, then uses self-supervised DTAE [48] for pre-training before applying downstream fine-tuning tasks. We experiment with classification loss (cross-entropy loss: ce) and representation loss (triplet loss: triplet) as loss functions in the fine-tuning network for the OSR tasks. To demonstrate that our proposed approach is effective for OSR problems, we compare our approach with OpenMax[4]. Moreover, to prove that the self-supervised pre-training step benefits the OSR tasks, we compare the results of using and not using self-supervised pre-training for the two types of loss functions mentioned above.

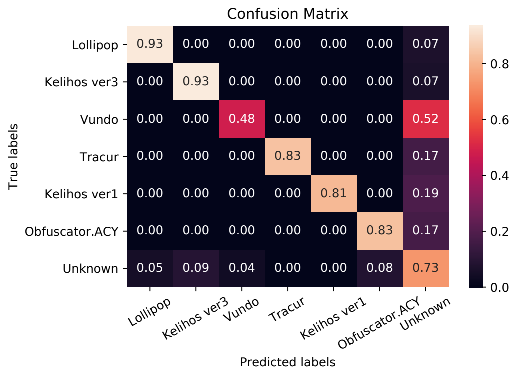
As illustrated in Figure 4.1a, the pre-trained network contains an encoder and a decoder. Furthermore, the learned encoder is fine-tuned with down-stream OSR tasks. For the encoder, the padded input layer is of size (67,67) for both MC and AG datasets. Two non-linear convolutional layers follow the input layer with 32 and 64 nodes. We apply the max-pooling layers with kernel size (3, 3) and strides (2, 2) as well as batch normalization after each convolutional layer. After a convolutional block, we add one fully connected non-linear layer with 256 hidden units before the representation layer, containing six dimensions. Moreover, We use the Relu activation function and set the Dropout rate as 0.2. We use Adam optimizer with a 0.001 learning rate. The decoder in the pre-trained network is simply the reverse of the encoder in our experiments. The encoder and representation layer maintain the same architecture and hyperparameters in the fine-tuning network. Meanwhile, the decoder is replaced with different fully connected layers associated with different loss functions.



(a) Without pre-training (ce)

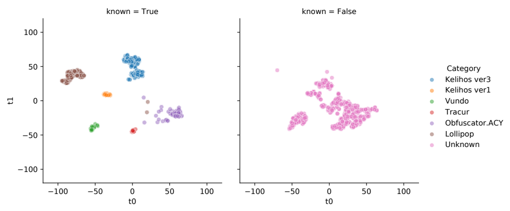


(b) After pre-training (FCG-random)

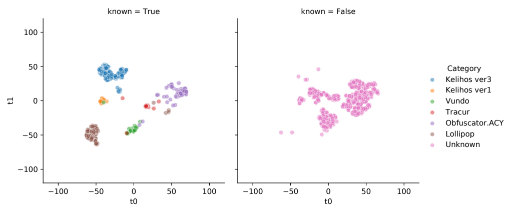


(c) After fine-tuning (FCG-random+ce)

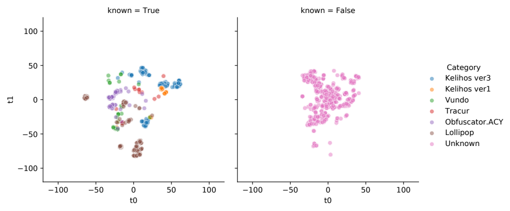
Figure 5.3: The confusion matirces of the MC test dataset under different settings: (a) Cross-entropy loss without pre-training; (b) Augmented with FCG-random and pre-trained with DTAE; (c) Fine-tuned with cross-entropy loss after (b).



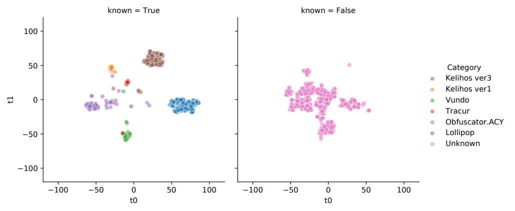
(a) OpenMax



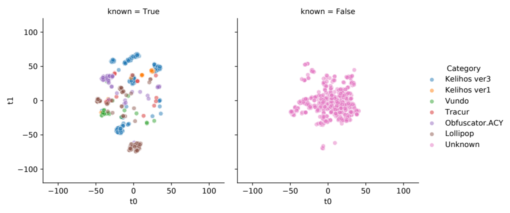
(b) Without pre-training (ce)



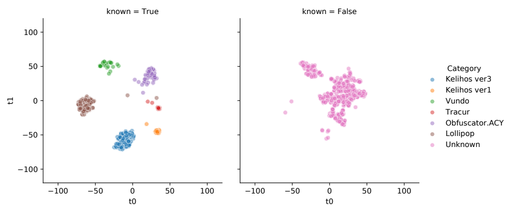
(c) After pre-training (Node dropping)



(d) After fine-tuning (Node dropping+ce)



(e) After pre-training (FCG-random)



(f) After fine-tuning (FCG-random+ce)

Figure 5.4: The t-SNE plots of the MC test representations learned by different settings: (a) OpenMax; (b) Cross-entropy loss without pre-training; (c) Augmented with node dropping and pre-trained with DTAE; (d) fine-tuned with cross-entropy loss after (c); (e) Augmented with FCG-random and pre-trained with DTAE; (f): fine-tuned with cross-entropy loss after (d). The left subplots are the representations of the known class, and the right subplots are the representations of the unknown classes.

Table 5.3: The average F1 scores of 30 runs of OpenMax and a group of 6 methods (without pre-training using manually selected threshold as baseline, without pre-training using statistical threshold, pre-training via DTAE with transformations node dropping, subgraph sampling, FCG-shift and FCG-random) for each of the two types of loss functions (ce and triplet). The values in bold are the highest values in each group. The underlined values are statistical significant better than OpenMax.

		MC			AG		
		Known	Unknown	Overall	Known	Unknown	Overall
OpenMax		$0.891_{\pm0.006}$	$0.737_{\pm0.010}$	$0.869_{\pm0.006}$	$0.408_{\pm0.190}$	$0.640_{\pm0.163}$	$0.441_{\pm0.184}$
ce	No pre-training (manual threshold)	$\mathbf{0.899_{\pm0.010}}$	$0.703_{\pm0.061}$	$0.871_{\pm0.017}$	$0.683_{\pm0.117}$	$0.540_{\pm0.329}$	$0.663_{\pm0.120}$
	No pre-training (statistical threshold)	$0.890_{\pm0.021}$	$0.663_{\pm0.176}$	$0.858_{\pm0.042}$	$0.705_{\pm0.088}$	$0.512_{\pm0.363}$	$0.678_{\pm0.120}$
	Node dropping	$0.852_{\pm0.077}$	$0.715_{\pm0.097}$	$0.833_{\pm0.078}$	$0.684_{\pm0.176}$	$0.636_{\pm0.339}$	$0.677_{\pm0.181}$
	Subgraph sampling	$0.000_{\pm0.000}$	$0.384_{\pm0.000}$	$0.055_{\pm0.000}$	$0.006_{\pm0.018}$	$0.616_{\pm0.210}$	$0.093_{\pm0.016}$
	FCG-shift (ours)	$\underline{0.896_{\pm0.010}}$	$\underline{0.765_{\pm0.024}}$	$\underline{0.878_{\pm0.011}}$	$\underline{\mathbf{0.743_{\pm0.088}}}$	$\mathbf{0.612_{\pm0.327}}$	$\underline{\mathbf{0.724_{\pm0.113}}}$
	FCG-random (ours)	$\underline{0.898_{\pm0.012}}$	$\underline{\mathbf{0.774_{\pm0.025}}}$	$\underline{\mathbf{0.880_{\pm0.013}}}$	$\underline{0.647_{\pm0.129}}$	$0.608_{\pm0.318}$	$\underline{0.641_{\pm0.127}}$
triplet	No pre-training (manual threshold)	$0.905_{\pm0.007}$	$0.728_{\pm0.035}$	$0.879_{\pm0.011}$	$0.753_{\pm0.074}$	$0.789_{\pm0.133}$	$0.758_{\pm0.068}$
	No pre-training (statistical threshold)	$0.903_{\pm0.010}$	$0.749_{\pm0.036}$	$0.881_{\pm0.013}$	$0.771_{\pm0.059}$	$\mathbf{0.827_{\pm0.093}}$	$0.779_{\pm0.054}$
	Node dropping	$0.884_{\pm0.036}$	$0.736_{\pm0.046}$	$0.862_{\pm0.037}$	$0.679_{\pm0.184}$	$0.768_{\pm0.170}$	$0.692_{\pm0.171}$
	Subgraph sampling	$0.014_{\pm0.075}$	$0.372_{\pm0.069}$	$0.065_{\pm0.054}$	$0.011_{\pm0.061}$	$0.657_{\pm0.135}$	$0.104_{\pm0.036}$
	FCG-shift (ours)	$\underline{\mathbf{0.906_{\pm0.007}}}$	$\underline{0.758_{\pm0.021}}$	$\underline{\mathbf{0.885_{\pm0.008}}}$	$\underline{0.745_{\pm0.074}}$	$0.744_{\pm0.250}$	$\underline{0.745_{\pm0.092}}$
	FCG-random (ours)	$\underline{\mathbf{0.906_{\pm0.007}}}$	$\underline{\mathbf{0.763_{\pm0.020}}}$	$\underline{\mathbf{0.885_{\pm0.008}}}$	$\underline{\mathbf{0.776_{\pm0.061}}}$	$\underline{0.819_{\pm0.166}}$	$\underline{\mathbf{0.782_{\pm0.067}}}$

5.3.2 Evaluation Criteria

To simulate an open-set scenario, we randomly pick six out of nine classes as the known classes and used them in training, and samples from the other classes are regarded as the unknown class, which only exists in the test set. We simulate three different open set groups for each dataset and then repeat each group 10 runs, so each dataset has 30 runs. We calculate the average results of 30 runs for performance evaluation.

We perform a three-dimensional comparison for our proposed approach. First, to show that our proposed approach can achieve good performance in the OSR problem, we compare our proposed approach with the popular OSR solution OpenMax [4]. Moreover, to verify that the self-supervised pre-training process benefits the OSR problem for different downstream loss functions, we compare the model performances with and without using the pre-training process. Finally, we compare our proposed transformation methods

"FCG-shift" and "FCG-random" with other graph transformations "Node dropping" (ND) and "Subgraph sampling" (SS), which are generally beneficial across datasets [100]. While the AUC score under 100% FPR is commonly used in model performance measurements, the AUC score under 10% FPR is more meaningful for malware detection applications. Moreover, we measure the F1 scores for classifying the known classes correctly and recognizing the unknown class correctly for the OSR system. Finally, to show that our proposed statistical approach to recognizing unknown classes in Section 5.2.5 performs as good as the manual thresholding approach: sort the outlier score of the training date in ascending order and then manually pick an outlier score value (99 percentile) as the outlier threshold as in [37, 48], we compare two different thresholding strategies – "manual threshold" and "statistical threshold" – on the representations learned by the vanilla models without pre-training process. To verify that our proposed approaches achieve significant improvement on the OSR, we perform t-tests against OpenMax with 95% confidence in both the AUC scores and F1 scores.

5.3.3 Experimental Results

We test our proposed pre-training strategy on downstream networks with classification (cross-entropy loss) and representation (triplet loss) loss functions and apply the statistical thresholding approach to learned representations. Table 5.2 shows the average ROC AUC scores of the model performances in two malware datasets under different FPR values: 100% and 10%. Comparing "ce" and "triplet" columns with "OpenMax" columns, we observe that no matter with or without our proposed pre-training process, the models that use cross-entropy loss and triplet loss perform better than OpenMax for our malware datasets. Furthermore, our proposed pre-training approach outperforms the models without the pre-training process in all 8 cases (2 datasets $\times$ 2 FPRs $\times$ 2 loss functions). On the contrary, the DTAE pre-training with node dropping transformation does not benefit the model performance, and the subgraph sampling transformation even hurts the model performance. For MC dataset, the FCG-random transformation works better than the FCG-shift transformation. Meanwhile, their performances differ with different loss functions for the AG dataset.

We also measure the OSR performances via F1 scores under different categories. As shown in Table 5.3. The three categories are: "Known", "Unknown", and "Overall". Specifically, the "Known" category is the average F1 scores of the known classes. Moreover, the "Overall" category is the average F1 scores of the known and unknown classes. We observe that the pre-training with our proposed transformation methods improves the model performances in the majority of the cases. However, the pre-training with node dropping and subgraph sampling hurts the model performance in most cases. Moreover, the results in the "manual threshold" and "statistical threshold" rows indicate that our proposed statistical thresholding strategy in Section 5.2.5 can achieve similar performance with the manually selected threshold. Meanwhile, the statistical thresholding approach reduces the number of hyperparameters and alleviates the grid searching process.

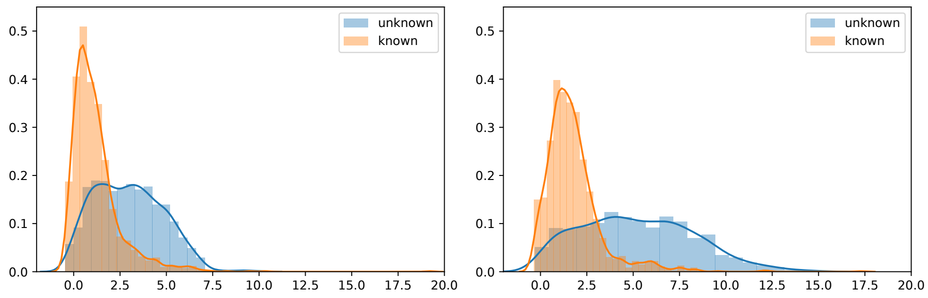
Overall, we notice that for both ROC AUC scores and F1 scores, the DTAE pre-training using our proposed transformation approach benefits the model performance in OSR problems. Meanwhile, the transformation method node dropping does not help malware FCGs datasets. As discussed in Section 5.2.3, the FCGs are, in general, very sparse graphs. Dropping nodes and subgraph sampling will potentially lose important information about the malware. Meanwhile, our proposed FCG-shift and FCG-random transformation will preserve all the information by creating isomorphic views.

5.3.4 Analysis

While the ROC AUC and F1 scores show that our proposed pre-training approach improves the models' performances, we plot the confusion matrices of one set of the experiments with the MC test set to analyze the experiment results further. In the experiments, the known malware classes are "Lollipop", "Kelihos ver3", "Vundo", "Tracur", "Kelihos ver1", and "Obfuscator.ACY", the remaining three classes together are considered as the unknown class not participating in the training process. Figure 5.3a shows the confusion matrix of the model using cross-entropy without pre-training. Figure 5.3b and Figure 5.3c are the confusion matrices of the model performance after pre-training with FCG-random and after being fine-tuned with cross-entropy loss, respectively. According to the true positive (TP) predictions along the diagonals of the confusion matrices in Figure 5.3b, the model can already classify the known classes after the pre-training stage. Comparing the model performance without pre-training in Figure 5.3a and the one with pre-training in 5.3c, we observe that the TP predictions have been significantly increased for the unknown class. While the TP predictions on the "Vundo" class have decreased, the False Positive (FP) predictions (off-diagonal values) happen only between the known classes and the unknown class instead of among the known classes, which indicates that the known classes are more separable.

To visualize the differences between learned representations, we generate the t-SNE plots of the representations at different stages in different experiments as in Figure 5.4. Specifically, Figure 5.4a is the t-SNE plot of the learned representations of OpenMax. Figure 5.4b shows the representations learned by the model using cross-entropy loss without pre-training. Figures 5.4c and 5.4d are the representations learned by the model after pre-training with node dropping and being fine-tuned with cross-entropy loss. Figure 5.4e and Figure 5.4f are the representations learned by pre-trained model using DTAE with FCG-random and after being fine-tuned with cross-entropy loss. From the left subplot in Figure 5.4c and Figure 5.4e, we observe that even without class label information, the self-supervised pre-training model can capture some cluster information. We can find the tiny clusters for the "Obfuscator.ACY" class, "Kelihos ver3" class and "Lollipop" class, which explains the behavior in Figure 5.4c and Figure 5.3b. Moreover, in Figure 5.4f, the representations of the known classes in the left subplots are more separate from each other. Meanwhile, the representations of the unknown class are more concentrated near the origin.

Figure 5.5 shows the distributions of the average outlier scores for the known and unknown classes for the MC test set. Comparing the distributions of outlier scores generated from cross-entropy loss without pre-training in Figure 5.5a and with pre-training in Figure 5.5b, we notice that while the pre-training process increases the outlier scores for both the known classes and the unknown class, it increases the outlier scores in the unknown classes more significantly, which pushes the distribution further away from the known classes. Therefore, there is less overlap and higher accuracy.



(a) Without pre-training (ce)

(b) With pre-training (FCG-random + ce)

Figure 5.5: The distributions of outlier scores for the known and unknown classes of the MC dataset using cross-entropy loss with and without pre-training process.

5.4 Conclusion

In this chapter, we design a two-stage learning process for learning the representations of the malware FCGs to resolve the set recognition problem of malware samples. Specifically, we propose two transformation methods for the FCGs to facilitate the detransformation autoencoder (DTAE) in the pre-training step. Then, we fine-tune the network with different types of loss functions. Moreover, to find the optimal threshold for the OSR problem, we design a statistical thresholding approach based on the distribution of learned representations. The proposed approach reduced the number of hyper-parameters and hence the costs of the resources for the hyperparameter tuning process. We evaluate the pre-training approach with classification loss and representation loss functions on two malware datasets. The results indicate that our proposed approach can improve both model performances for the OSR tasks.

Chapter 6

Feature Decoupling in Self-supervised Representation Learning for Open Set Recognition

6.1 Introduction

As classification techniques have achieved great success in various fields in research and industry, most traditional classification problems focus on the known classes. However, collecting samples exhausting all classes in the real world is difficult. This problem is referred as Open Set Recognition (OSR) [4]. OSR attempts to handle the known classes that already exist in the training set and the unknown classes that are absent from the training set. Hence, for a multinomial classification problem, an OSR task normally involves two objectives: to classify the known classes and reject the unknown class.

In this chapter, we introduce a two-stage learning strategy for the OSR problem. The first stage extracts the content features via a self-supervised learning approach. The majority of the self-supervised learning methods focus on designing various pre-text tasks [27, 102, 48]. These pretext tasks usually aim to learn content features, which implicitly try to remove the transformation information. We hypothesize that explicitly learning separate content and transformation features can improve the content features. During the first (pre-training) stage, we introduce a feature decoupling approach to extract the content features irrelevant to transformation information. The proposed approach decouples the representations through content reconstruction and transformation prediction tasks. A learned representation in our network is a concatenation of content features and transformation features. The content features are irrelevant to the transformations. Thus, the content features of the different views from the same original input should be the same. We achieve this goal by reconstructing the different views to their original form. Furthermore, the transformation features should contain the discriminative information on the transformation types. Thus, we introduce the transformation labels inside the input transformation module and build an auxiliary transformation classifier on top of the transformation features. After explicitly learning separate content and transformation features, in the second stage, we fine-tune the content features with the provided class labels and discard the transformation features. We look into two different supervised loss functions: classification loss and representation loss. The classification loss, such as cross-entropy loss, is applied to the decision layer to lower the classification error. The representations loss, such as triplet loss [80], is applied directly to the representation layer to decrease the intra-class spread and increase inter-class separation. In the case of classification loss, we connect a content classifier to the content features and fine-tune the network with a content classification loss. In the case of representation loss, we apply the loss function directly to the content features. Finally, the fine-tuned content features are used for the OSR tasks. We also consider an unsupervised learning scenario in the second stage, where the known class labels are unavailable. In this case, we cluster the content features learned in the first stage to find the potential classes.

Our contribution includes: first, we design a two-stage training strategy for the OSR tasks. Among these, we propose a feature decoupling approach to extract the content features that are irrelevant to the transformation information. Second, we extend our approach to the unsupervised scenario in OSR. Third, to evaluate the quality of learned representations of the pre-training and fine-tuning stages, we propose intra-inter ratio (IIR) and show that it is correlated to OSR performance. Lastly, we experiment with different loss functions with image and malware datasets. The results indicate that our proposed self-supervised learning method is more effective than other approaches in OSR.

We organize this chapter as follows. Section 6.2 presents a two-stage training strategy for learning content features and introduces how to use the learned content features to perform the OSR tasks. In Section 6.3, we evaluate our proposed approach through experiments on different types of datasets and also compare the experimental results with other approaches.

6.2 Approach

In this section, we first describe a two-stage learning process to learn the representations of input samples. In the first stage (pre-training), we utilize a self-supervision approach to extract the low-level content features of the input samples. In the second stage (fine-tuning), we introduce two types of loss functions (classification loss and representation loss) to fine-tune the discriminative content features. Then, we present a recognition strategy for the OSR problem with the centroids of the fine-tuned content features. Moreover, we consider an unsupervised scenario for the second stage of learning, where the labels of known classes are unavailable. In this case, we cluster the learned content features with K-Means instead of fine-tuning them with class labels in the second stage. Moreover, cluster centroids are used in the recognition strategy for the OSR problem.

6.2.1 Pre-training stage: self-supervised feature decoupling

Self-supervised learning uses pretext tasks in the objectives, which generally incorporate the transformations of original input samples. Thus, the learned features contain two types of information for a transformed input sample: transformation unrelated content information and transformation information. The information solely related to transformation is introduced by pretext tasks, which are not practical for the downstream tasks. Therefore, we develop a feature decoupling method to separate these two types of information into transformation unrelated content features and transformation features.

As shown in Figure 6.1, given an original input sample x, we use a transformation module T augments the input x with different several correlated views. The transformation module contains M different transformations as $T = \{t_1, t_2, ...t_m\}$. In our example in Figure 6.1, the transformations are the rotations of input image samples by multiples of 90 degrees (0°, 90°, 180° and 270°) such that $M = 4$. We denote the original input x with transformation t_j by x_j. i.e, $x_j = t_j(x)$. Then, a network-based encoder $f(\cdot)$ extracts the representation vector z_j from transformed data example x_j, such that $z_j = f(x_j)$. We suppose that the high level representation vector z_j can be represented as $z_j = [z_j^c, z_j^t]$, where z_j^c is content features of the transformed data example while z_j^t is responsible for the transformation features. We apply two different objectives to decouple these two types of information. Specifically, we use a reconstruction decoder $g^c(\cdot)$ to learn the content features and a transformation classifier $g^t(\cdot)$ to extract the transformation related features.

6.2.1.1 Learning content features

The content features should be invariant to the transformations. In other words, for the same input sample, the content features should be invariant for all its transformed views. SimCLR [12] and Barlow Twins [102] achieve such agreement by maximizing similarity of representations obtained from different transformed views of a sample in the latent space. DTAE [48] encourages the similarity of representations of different views by reconstructing them back to their original forms. Here, we apply a content reconstruction decoder on reconstructed transformed samples. As shown in the "Objective 1" in Figure 6.1, the input of the decoder is the content part of the representation, z_j^c. Moreover, instead of reconstructing the content features back

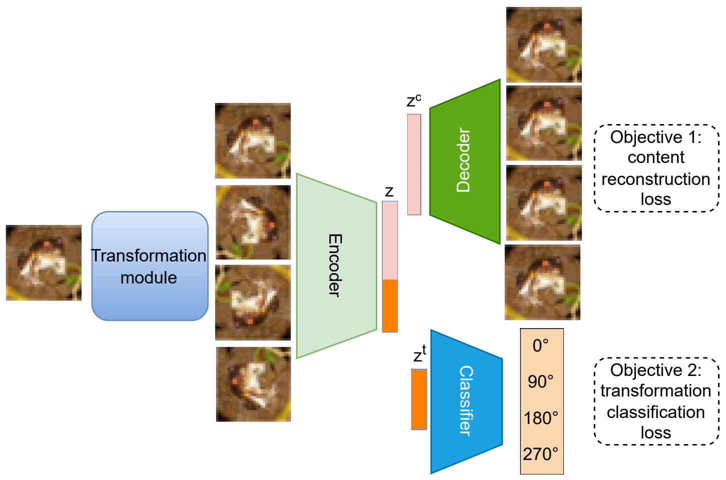

Transformation module
Encoder
z
z^c
z^t
Decoder
Classifier
0°
90°
180°
270°
Objective 1: content reconstruction loss
Objective 2: transformation classification loss

Figure 6.1: Illustration of proposed feature decoupling method. The transformation module transforms the original input samples into several correlation views. The encoder outputs decoupled content and transformation features. The content part is learned by reconstructing the transformed input samples back to their original forms, and the transformation part is learned by transformation classification.

to their transformed views, the decoder here "reconstructs" them to their original form before the transformation module.

Specifically, let $g^c(z^c_j)$ denotes reconstruction from the content feature of the transformed view x_j, we use MSE (Mean Squared Error) loss to maximize the similarity of the reconstruction and the original input sample x:

$$\mathcal{L}_{\text{content}} = \frac{1}{2}\sum_{j=1}^{M}(x - g^c(z^c_j))^2 \tag{6.1}$$

Where each of the data points has M transformations, and there are M times data points as the original input sample after the transformation module.

6.2.1.2 Learning transformation features

Towards the goal of extracting transformation features, we apply a classifier to predict the transformation classes introduced from the transformation module. As shown in the "Objective 2" in Figure 6.1, the input of the transformation classifier is the transformation part of the representation, z_j^t, and the output is the prediction logits of transformation classes, i.e., the rotation angles in our example. Formally, given the transformation part of the representation z_j^t, the classifier outputs the transformation prediction logit of the i-th class: $p_i(g^t(z_j^t))$. We use a softmax cross-entropy loss for the transformation classification and write the loss functions as:

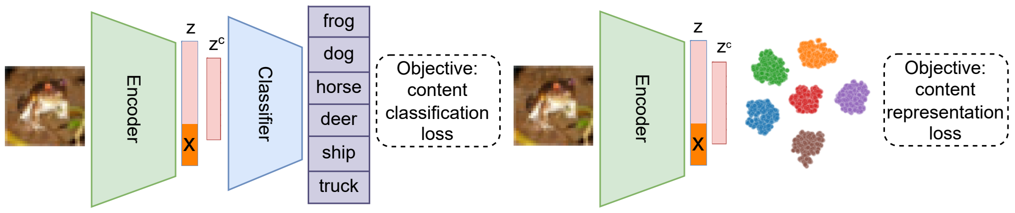
$$\mathcal{L}_{\text{transformation}} = -\log p_{i=j}(g^t(z_j^t)), \tag{6.2}$$

where j is the ground truth transformation label of input sample x_j. In our example in Figure 6.1, the objective of the classifier is classifying four rotation types that introduced from the transformation module.

Algorithm 2 Pre-training stage of feature decoupling in self-supervised representation learning

```
    Input: Training data and labels $(x, y)$.
    Output: Encoder $f(\cdot)$, content decoder $g^c(\cdot)$, and
            transformation classifier $g^t(\cdot)$.
 1: Random Initialize $f(\cdot)$, $g^c(\cdot)$ and $g^t(\cdot)$;
 2: for each transformation $j$ do
 3:     $j, x_j \leftarrow t_j(x)$
 4: for epochs do
 5:     for each transformation $j$ do
 6:         Extract the representation $z_j[z_j^c, z_j^t]$ from $f(x_j)$;
 7:         Reconstruct the original input $x$ from $g^c(z_j^c)$;
 8:         Train $f(\cdot)$ and $g^c(\cdot)$ by Eq. 6.1;
 9:         Classify the transformation label $j$ from $g^t(z_j^t)$;
10:         Train $f(\cdot)$ and $g^t(\cdot)$ by Eq. 6.2;
    return $f(\cdot)$, $g^c(\cdot)$, $g^t(\cdot)$.
```

In Algorithm 2, we summarize the overall pre-training stage of the proposed feature decoupling method. After initializing the networks in Line 1 and transforming the original training data in Line 2, the network is trained


Encoder
z
z^c
X
Classifier
frog
dog
horse
deer
ship
truck
Objective: content classification loss
Encoder
z
z^c
X
Objective: content representation loss

(a) Fine-tuning with classification loss (b) Fine-tuning with representation loss

Figure 6.2: Illustration of two types of fine-tuning objectives. only the original input sample are used in the fine-tuning stage. The encoder and representation layer are inherited from the pre-training stage, then the content features (z^c, pink) are connected to (a) a classifier or (b) a representation loss function for fine-tuning. The transformation features (orange) are discarded.

using mini-batch stochastic gradient descent with backpropagation. During each epoch, we first extract the representation z_j (Line 4). Lines 6-7 train the encoder $f(\cdot)$ and content decoder $g^c(\cdot)$ under the guidance of content reconstruction loss function (Equation 6.1) on the content part of the representation z_j^c. Next, Line 8 and 9 execute the training for the transformation part of the representation z_j^t under the transformation classification loss function in Equation 6.2.

6.2.2 Fine-tuning stage: supervised fine-tuning

The self-supervised feature decoupling approach in the first stage attempts to find the low-level content features. We further fine-tune the content features learned in the first stage with the class labels. The fine-tuning stage incorporates the available class labels in the training data to discover the discriminative features between classes for class awareness. The loss functions of the fine-tuning network can be categorized into two types: classification and representation loss.

The classification loss function requires a classifier connected to the representation layer, which is applied to the output logits in the decision layer. One of the widely used classification loss functions is cross-entropy loss. Figure 6.2a illustrates the network architecture of using the classification loss in the fine-tuning step. Compared with the pre-training step, the fine-tuning step only uses the original training data. The training data is passed through the encoder learned in the pre-training step. Moreover, instead of connecting to a content decoder, the content part of the representation connects to a classifier that outputs class logits. The classification loss is applied to this output to lower the classification error.

Unlike the classification loss, the representation loss functions do not require a classifier. They constrain the representation layers directly, such as triplet loss [80]. Figure 6.2b illustrates how we incorporate the representation loss in the fine-tuning stage. After passing the input data through the pre-trained encoder, we extract the decoupled representations. Then, instead of a content decoder or a classifier, the content part of the representation is directly constrained by the representation loss function. After fine-tuning the encoder with the labeled dataset, we calculate centroid u_k of class k based on the content features:

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{N_k} z_i^c, \tag{6.3}$$

where N_k is the number of training instances in class k. During the inference time, we only use the content features z^c to represent the input sample for the OSR tasks.

6.2.3 Open set recognition

After the second stage, we obtain the encoder and centroids of all the known classes. We have two problems to solve for an OSR task: classifying the known classes and identifying the unknown class. If we have K known classes, given the content features z^c of test sample x, we define the outlier score as the Euclidean distance to its closest centroid:

$$outlier\_score(x) = \min_{1 \leq k \leq K} \|\mu_k - z^c\|_2^2 \tag{6.4}$$

In this work, the outlier threshold t is the 99 percentile of the outlier score in ascending order. A test sample is recognized as unknown if an outlier score exceeds the selected threshold. Otherwise, we use a class probability $P(y = k|x)$ to decide the test sample belongs to which known class. For the fine-tuning network with classification loss, we use the output probability in the decision layer as $P(y = k|x)$. For the fine-tuning network with representation loss, we calculate $P(y = k|x)$ as:

$$P(y = k|x) = \frac{e^{-\|\mu_k - z\|_2^2}}{\sum_{k=1}^{K} e^{-\|\mu_k - z\|_2^2}} \tag{6.5}$$

And the test sample is classified as the known class with the highest class probability.

$$\hat{y} = \begin{cases} unknown, & \text{if } outlier\_score(x) > t \\ \\ \underset{1 \leq k \leq K}{\operatorname{argmax}} P(y = k|x), & \text{otherwise} \end{cases} \tag{6.6}$$

6.2.4 Extension to unsupervised OSR

When the class labels of the known classes are unavailable, the problem becomes an unsupervised OSR problem. The unsupervised OSR aims to identify whether an instance is from the known data distributions or an unknown data distribution without known class labels. In this scenario, after self-supervised pre-training (Sec. 6.2.1), instead of supervised fine-tuning (Sec. 6.2.2), we apply a clustering algorithm (such as K-Means) to identify clusters based on the content features learned from pre-training. That is, we do not perform supervised fine-tuning of the features. After finding the clusters, we calculate the centroid of each cluster (instead of each class) according to Eq. 6.3. Centroids of the clusters (instead of the classes) are then used for OSR as discussed in Sec. 6.2.3. For reference convenience, we assign an ID to each cluster. Unsupervised OSR outputs one of the known cluster IDs or unknown.

6.3 Experiments

We evaluate the proposed feature decoupling approach with two types of fine-tuning functions as mentioned in Section 6.2.2: classification loss (cross-entropy loss) and representation loss (triplet loss). Moreover, to show that our proposed approach works on different datasets. We test the proposed approach on images and malware datasets. A web link will be provided for our implementation and datasets in the paper if it is published.

Fashion-MNIST [93] is associated with 10 classes of clothing images. It contains 60,000 training and 10,000 testing examples. In the Fashion-MNIST dataset, each example is a 28x28 grayscale image. To simulate an open-set dataset, we randomly pick six digits as the known classes, while the rest are treated as the unknown class for testing.

CIFAR-10 [56] 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. As the Fashion-MNIST datasets and the FCGs datasets only have one channel, for consistency, we first convert the color images to grayscale and randomly pick six classes out of the ten classes as the known classes. In contrast, the remaining classes are treated as the known class only existing in the test set.

Microsoft Challenge (MS) [74] contains disassembled malware samples from 9 families:“Ramnit”, “Lollipop”, “Kelihos ver3”, “Vundo”, “Simda”, “Tracur”, “Kelihos ver1”, “Obfuscator.ACY ” and “Gatak”. We use 10260 samples that can be correctly parsed then extracted their FCGs as in [39] for the experiment. We randomly pick six classes of digits as the known classes participant in the training, while the rest are considered as unknowns that only exist in the test set.

Android Genome (AG) consists of 1,113 benign android apps and 1,200 malicious android apps. Our colleague provides the benign samples, and the malicious samples are from [108]. We select nine families with a relatively larger size for the experiment to be fairly split into the training set and the test set. The nine families contain 986 samples in total. We first use [23] to extract the function instructions and then generated the FCGs as in [39]. Also, to simulate an open-set scenario, we randomly pick six digits as the known classes while considering the rest as the unknown class.

6.3.1 Implementation details and comparison methods

Our proposed training process consists of two stages. In the first stage, we compare our proposed self-supervised feature decoupling (FD) approach with other self-supervised learning approaches: RotNet [27], Barlow Twins [102], DTAE [48][46]. We construct a fine-tuning network in the second stage to refine the learned content features. We experiment with classification loss (cross-entropy loss: ce) and representation loss (triplet loss: triplet) as loss functions in the fine-tuning network. Furthermore, To demonstrate that our proposed approach is effective for OSR problems, we compare our approach with OpenMax[4].

6.3.1.1 Self-supervised feature decoupling

As illustrated in Figure 4.1a, the pre-training stage includes a transformation module to facilitate the pre-text task. For the image datasets, we use the rotation of a multiplier of 90 degrees (e.g., 0, 90, 180, 270 degrees) in the transformation module. As for the malware datasets. We first extract the FCGs of each sample, then apply FCG-random [46] on the FCGs. The transformed samples are then passed through an encoder. The padded input layer size varies for different datasets. For the Fashion-MNIST dataset, the input images are of size (28, 28) and are padded to get the size (32, 32) with one channel. For the CIFAR-10 dataset, the padded input size is (36, 36). For the FCG datasets (MS and Android), the padded input layer is in the size of (67, 67). The padded input layer is then flowed by two non-linear convolutional layers with 32 and 64 nodes. We apply the max-polling layers with kernel size (3, 3) and strides (2, 2). We also add batch normalization after each convolutional layer to complete the convolutional block. After the convolutional block, we use two fully connected non-linear layers with 256 and 128 hidden units for the image datasets Fashion-MNIST and CIFAR-10. We only use one fully connected non-linear layer with 256 hidden units for the graph dataset. Furthermore, the size of representations is nine dimensions for all the datasets in our experiments, with six dimensions for the content features and the remaining three for the non-content transformation features. The six-dimensional content features are connected to a decoder, which is simply the reverse of the encoder. The three-dimensional transformation features are further connected to a linear layer and then fed to a softmax layer for the transformation classification. We use the Relu activation function and set the Dropout rate as 0.2. We use Adam as the optimizer with a learning rate of 0.001.

The comparison methods RotNet, Barlow Twins, and DTAE share the same backbone encoder architecture as our proposed method. Their representation layers have six dimensions. Also, we have generalized the original RotNet and Barlow Twins methods for fair comparison in our experiments. Specifically, the pre-text task of original RotNet was classifying the rotation degrees in [27], which is only applicable for the image datasets. Here, to make RotNet feasible for the FCG datasets, we extend the pre-text task to predicting the FCG-random transformation labels for the FCG datasets. Moreover, the original Barlow Twins impose constraints on the cross-correlation matrix between the representations of two transformed views in [102]. Here, we apply the same constraints to the cross-correlation matrices between the transformed views and their corresponding original samples.

Table 6.1: The average ROC AUC scores of 30 runs at 100% and 10% FPR of OpenMax and a group of 5 methods (without pre-training, pre-training with RotNet, Barlow Twins, DTAE and Feature Decoupling (FD)) for two loss functions: cross-entropy loss and triplet loss under supervised OSR scenario. The values in bold are the highest values in each group.

		Fashion-MNIST		CIFAR-10		MS		AG	
	FPR	100%	10%	100%	10%	100%	10%	100%	10%
OpenMax		0.740$_{\pm 0.046}$	0.016$_{\pm 0.008}$	0.675$_{\pm 0.017}$	0.006$_{\pm 0.001}$	0.880$_{\pm 0.037}$	0.040$_{\pm 0.002}$	0.480$_{\pm 0.190}$	0.001$_{\pm 0.001}$
ce	No Pre-training	0.717$_{\pm 0.036}$	0.029$_{\pm 0.005}$	0.580$_{\pm 0.046}$	0.007$_{\pm 0.001}$	0.914$_{\pm 0.030}$	0.052$_{\pm 0.006}$	0.853$_{\pm 0.082}$	0.022$_{\pm 0.014}$
	RotNet	0.736$_{\pm 0.047}$	0.031$_{\pm 0.007}$	0.612$_{\pm 0.040}$	0.008$_{\pm 0.001}$	0.911$_{\pm 0.032}$	0.055$_{\pm 0.005}$	0.870$_{\pm 0.059}$	**0.026**$_{\pm 0.017}$
	Barlow Twins	0.719$_{\pm 0.034}$	0.028$_{\pm 0.007}$	0.606$_{\pm 0.017}$	0.007$_{\pm 0.001}$	0.915$_{\pm 0.022}$	0.053$_{\pm 0.003}$	0.850$_{\pm 0.068}$	0.020$_{\pm 0.011}$
	DTAE	0.748$_{\pm 0.040}$	0.032$_{\pm 0.006}$	0.618$_{\pm 0.019}$	0.008$_{\pm 0.001}$	0.941$_{\pm 0.018}$	**0.064**$_{\pm 0.002}$	0.855$_{\pm 0.079}$	0.023$_{\pm 0.013}$
	FD (ours)	**0.771**$_{\pm 0.032}$	**0.034**$_{\pm 0.006}$	**0.628**$_{\pm 0.012}$	**0.009**$_{\pm 0.001}$	**0.945**$_{\pm 0.010}$	0.060$_{\pm 0.002}$	**0.876**$_{\pm 0.047}$	0.025$_{\pm 0.013}$
triplet	No Pre-training	0.716$_{\pm 0.037}$	0.021$_{\pm 0.005}$	0.610$_{\pm 0.026}$	0.008$_{\pm 0.001}$	0.923$_{\pm 0.028}$	0.056$_{\pm 0.005}$	0.868$_{\pm 0.046}$	0.027$_{\pm 0.014}$
	RotNet	0.743$_{\pm 0.028}$	**0.025**$_{\pm 0.005}$	0.628$_{\pm 0.015}$	0.009$_{\pm 0.001}$	0.924$_{\pm 0.018}$	0.057$_{\pm 0.003}$	0.870$_{\pm 0.036}$	0.025$_{\pm 0.009}$
	Barlow Twins	0.709$_{\pm 0.041}$	0.021$_{\pm 0.007}$	0.621$_{\pm 0.016}$	0.009$_{\pm 0.001}$	0.918$_{\pm 0.018}$	0.054$_{\pm 0.003}$	0.871$_{\pm 0.035}$	0.022$_{\pm 0.006}$
	DTAE	0.744$_{\pm 0.028}$	0.023$_{\pm 0.003}$	0.632$_{\pm 0.015}$	0.009$_{\pm 0.001}$	0.928$_{\pm 0.017}$	**0.061**$_{\pm 0.002}$	**0.879**$_{\pm 0.030}$	**0.026**$_{\pm 0.010}$
	FD (ours)	**0.758**$_{\pm 0.030}$	**0.025**$_{\pm 0.004}$	**0.636**$_{\pm 0.016}$	**0.010**$_{\pm 0.001}$	**0.941**$_{\pm 0.014}$	**0.061**$_{\pm 0.003}$	0.876$_{\pm 0.029}$	0.025$_{\pm 0.011}$

6.3.1.2 Supervised fine-tuning

In the fine-tuning network, the encoder and representation layer maintains the same architectures as the pre-trained network. Then, instead of connecting the representation layer to a decoder and a transformation classifier, we only connect the content features of the representation layer to a decision layer (for classification loss) in Figure 4.1b or a representation loss function as shown in Figure 4.1c. Likewise, for the comparison methods, we connect their representation layers to the decision layer (for classification loss) or a representation loss function.

As one of the comparison methods, OpenMax does not have a pre-training stage. It shares the same encoder architecture as our backbone network, then the representation is directly fed to a softmax layer.

6.3.2 Evaluation criteria

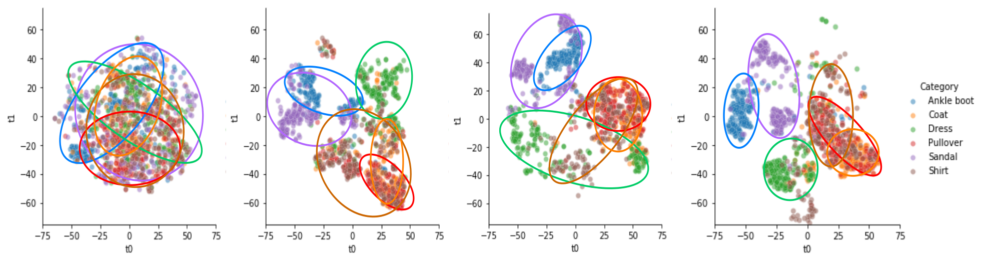
To simulate an open-set scenario, we randomly pick six classes as the known classes and use them in the training process. The ensemble of the remaining

Table 6.2: The average F1 scores of 30 runs OpenMax and a group of 5 methods (without pre-training, pre-training with RotNet, Barlow Twins, DTAE and Feature Decoupling) for two loss functions (cross entropy loss and triplet loss) under supervised OSR scenario. The values are the highest values in each group.

Image Dataset		Fashion-MNIST			CIFAR-10		
		Known	Unknown	Overall	Known	Unknown	Overall
OpenMax		$0.747_{\pm 0.049}$	$0.521_{\pm 0.178}$	$0.714_{\pm 0.051}$	$0.645_{\pm 0.022}$	$0.540_{\pm 0.065}$	$0.630_{\pm 0.017}$
ce	No Pre-training	$0.685_{\pm 0.102}$	$0.559_{\pm 0.076}$	$0.667_{\pm 0.086}$	$0.567_{\pm 0.048}$	$0.369_{\pm 0.169}$	$0.538_{\pm 0.045}$
	RotNet	$0.711_{\pm 0.067}$	$0.569_{\pm 0.097}$	$0.691_{\pm 0.058}$	$0.561_{\pm 0.061}$	$0.472_{\pm 0.136}$	$0.548_{\pm 0.049}$
	Barlow Twins	$0.738_{\pm 0.025}$	$0.506_{\pm 0.066}$	$0.704_{\pm 0.025}$	$\mathbf{0.599}_{\pm 0.022}$	$0.395_{\pm 0.105}$	$0.570_{\pm 0.023}$
	DTAE	$0.733_{\pm 0.050}$	$0.570_{\pm 0.087}$	$0.710_{\pm 0.041}$	$0.591_{\pm 0.037}$	$0.472_{\pm 0.096}$	$0.574_{\pm 0.027}$
	FD (ours)	$\mathbf{0.748}_{\pm 0.024}$	$\mathbf{0.587}_{\pm 0.075}$	$\mathbf{0.725}_{\pm 0.022}$	$0.587_{\pm 0.031}$	$\mathbf{0.514}_{\pm 0.067}$	$\mathbf{0.576}_{\pm 0.025}$
triplet	No Pre-training	$0.749_{\pm 0.014}$	$0.505_{\pm 0.075}$	$0.714_{\pm 0.021}$	$0.579_{\pm 0.042}$	$0.451_{\pm 0.134}$	$0.561_{\pm 0.038}$
	RotNet	$0.751_{\pm 0.015}$	$0.537_{\pm 0.075}$	$0.720_{\pm 0.020}$	$0.603_{\pm 0.033}$	$0.497_{\pm 0.087}$	$0.588_{\pm 0.030}$
	Barlow Twins	$0.740_{\pm 0.015}$	$0.433_{\pm 0.042}$	$0.696_{\pm 0.016}$	$0.609_{\pm 0.025}$	$0.446_{\pm 0.104}$	$0.586_{\pm 0.026}$
	DTAE	$\mathbf{0.755}_{\pm 0.010}$	$0.545_{\pm 0.076}$	$0.725_{\pm 0.018}$	$\mathbf{0.620}_{\pm 0.027}$	$0.472_{\pm 0.086}$	$0.599_{\pm 0.028}$
	FD (ours)	$0.753_{\pm 0.011}$	$\mathbf{0.582}_{\pm 0.083}$	$\mathbf{0.729}_{\pm 0.018}$	$0.617_{\pm 0.030}$	$\mathbf{0.515}_{\pm 0.028}$	$\mathbf{0.603}_{\pm 0.026}$
Malware Dataset		MS			AG		
		Known	Unknown	Overall	Known	Unknown	Overall
OpenMax		$0.891_{\pm 0.006}$	$0.737_{\pm 0.010}$	$0.869_{\pm 0.006}$	$0.408_{\pm 0.190}$	$0.640_{\pm 0.163}$	$0.441_{\pm 0.184}$
ce	No Pre-training	$0.899_{\pm 0.010}$	$0.703_{\pm 0.061}$	$0.871_{\pm 0.017}$	$0.683_{\pm 0.117}$	$0.540_{\pm 0.329}$	$0.663_{\pm 0.120}$
	RotNet	$0.900_{\pm 0.012}$	$0.708_{\pm 0.077}$	$0.872_{\pm 0.021}$	$0.709_{\pm 0.121}$	$\mathbf{0.613}_{\pm 0.335}$	$0.695_{\pm 0.135}$
	Barlow Twins	$0.896_{\pm 0.007}$	$0.712_{\pm 0.039}$	$0.870_{\pm 0.011}$	$0.701_{\pm 0.093}$	$0.541_{\pm 0.309}$	$0.678_{\pm 0.113}$
	DTAE	$\mathbf{0.908}_{\pm 0.008}$	$\mathbf{0.779}_{\pm 0.027}$	$\mathbf{0.890}_{\pm 0.010}$	$0.686_{\pm 0.107}$	$0.535_{\pm 0.280}$	$0.664_{\pm 0.110}$
	FD (ours)	$0.905_{\pm 0.007}$	$0.771_{\pm 0.026}$	$0.886_{\pm 0.009}$	$\mathbf{0.711}_{\pm 0.096}$	$0.612_{\pm 0.339}$	$\mathbf{0.697}_{\pm 0.118}$
triplet	No Pre-training	$0.905_{\pm 0.007}$	$0.728_{\pm 0.035}$	$0.879_{\pm 0.011}$	$0.753_{\pm 0.074}$	$0.789_{\pm 0.133}$	$0.758_{\pm 0.068}$
	RotNet	$0.906_{\pm 0.008}$	$0.739_{\pm 0.031}$	$0.882_{\pm 0.011}$	$0.755_{\pm 0.069}$	$0.791_{\pm 0.178}$	$0.760_{\pm 0.074}$
	Barlow Twins	$0.896_{\pm 0.006}$	$0.699_{\pm 0.034}$	$0.868_{\pm 0.010}$	$\mathbf{0.761}_{\pm 0.081}$	$0.739_{\pm 0.247}$	$0.757_{\pm 0.091}$
	DTAE	$\mathbf{0.911}_{\pm 0.006}$	$0.751_{\pm 0.024}$	$\mathbf{0.889}_{\pm 0.009}$	$0.734_{\pm 0.079}$	$0.735_{\pm 0.197}$	$0.734_{\pm 0.081}$
	FD (ours)	$0.909_{\pm 0.007}$	$\mathbf{0.762}_{\pm 0.031}$	$\mathbf{0.889}_{\pm 0.010}$	$0.760_{\pm 0.059}$	$\mathbf{0.807}_{\pm 0.160}$	$\mathbf{0.766}_{\pm 0.061}$

classes is considered the unknown class, which does not participate in the training process and only exists in the test set. We experiment with both supervised and unsupervised OSR scenarios.

For the supervised scenario, we simulate three groups of such open sets and experiment with each group with ten runs. We calculate the average results of these 30 runs when evaluating the model performances. For evaluation, we perform a three-dimensional comparison of our proposed approach. First, we compare model performances with and without using the pre-training process to verify that the pre-training process benefits the OSR problem for different loss functions. Second, we compare our feature decou-



(a) RotNet (b) Barlow Twins (c) DTAE (d) Feature Decoupling

Figure 6.3: The t-SNE plots of the representations of Fashion-MNIST test samples on pre-trained models.

pling (FD) approach with other self-supervised pre-training approaches, RotNet, Barlow Twins, and DTAE. Finally, to show that the two-stage trained model can achieve good performance compared to other OSR approach, we compare the proposed approach with the popular OSR solution OpenMax. Similar to the unsupervised scenario, we measure both ROC AUC scores under 100% and 10% FPRs. The ROC AUC scores under 100% FPR is commonly used in measuring model performance. However, in real-life applications such as malware detection, a lower FPR is more desirable. Thus the ROC AUC scores under 10% FPR are more meaningful in these cases. Moreover, as the objective of the OSR problem is twofold: classifying the known classes and recognizing the unknown class, we evaluate the F1 scores for the known class and the unknown class separately.

6.3.3 Evaluation results

Table 6.1 reports the AUC ROC scores under different FPR values: 100% and 10% under the supervised OSR scenario, where the known class labels are available. Comparing the "No Pre-training" rows with "OpenMax" rows of both loss functions, we observe that without the pre-training stage, OpenMax

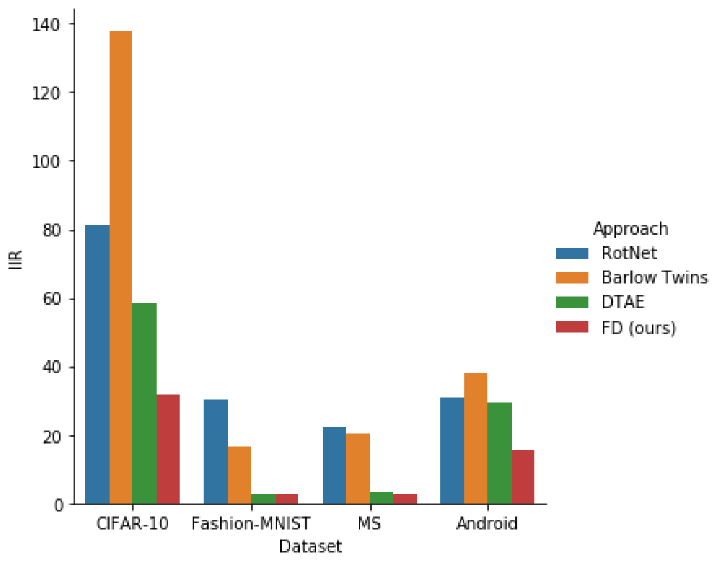


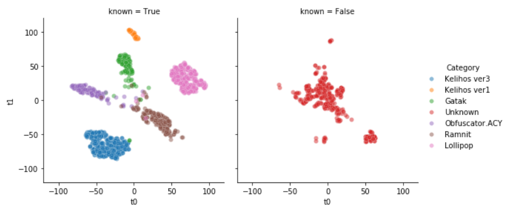
Figure 6.4: IIR after the pre-training stage.

outperforms the cross-entropy loss and triplet loss in the image datasets. On the contrary, for the malware datasets, the cross-entropy and triplet loss perform better than OpenMax. Moreover, comparing the models without pre-training stages with those with pre-training stages, we observe that the pre-training methods benefit the model performances in most cases. Also, the model pre-trained with our proposed feature decoupling (FD) achieves the best performance in 12 out of 16 comparison groups (4 datasets x 2 FPRs x 2 loss functions). Especially, the model pre-trained with our proposed approach achieves the best performance in all the cases in the graph datasets.

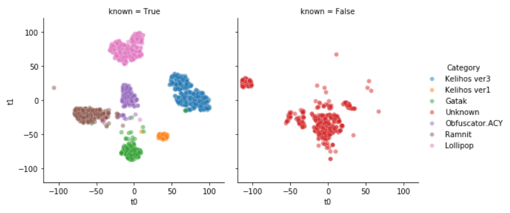
Besides the AUC ROC scores, we measure the F1 scores of different methods in Table 6.2. Notably, we measure the performance under three categories: the average F1 scores of all the known class (”Known” columns), the F1 scores of the unknown class (”Unknown” columns), and the average F1 scores of the known and unknown classes (”Overall” columns). Similar to the AUC ROC results, OpenMax outperforms cross-entropy loss and triplet loss in the image datasets when no pre-training stage is involved. However, both loss functions surpass OpenMax in the malware datasets. Moreover, all the pre-training methods benefit the model performance in classifying the known classes and recognizing the unknown class in most cases. Our proposed approach outperforms the other pre-training methods in 15 out of 24 groups (4 datasets x 3 categories x 2 loss functions). Especially for the "Overall" performances, our proposed approach achieves the best performance in 7 out of 8 groups.

From the experiment results, we observe that the performance of OpenMax differs on image and malware datasets. Also, a pre-training stage boosts the model performance on both classification and representation loss. Moreover, in most cases, our proposed self-supervised feature decoupling approach outperforms the other pre-training methods.

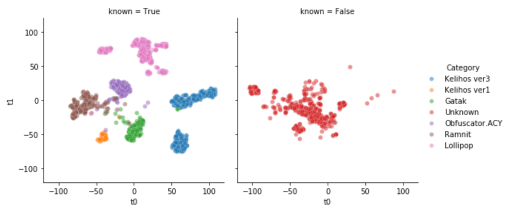
We perform an ablation study from two perspectives for our approach. First, from the two-stage training perspective, we study the effects of the pre-training stage. We compare the AUC scores in the “No Pre-training” rows and “FD(ours)” rows in Table 6.1 and Table 6.2. As expected, we observe that the pre-training stage has played an important role in the process. Second, our proposed pre-training approach, Feature Decoupling, has two components: a content reconstructor and a transformation classifier. The content reconstructor shares the same objective as DTAE. The results in the “DTAE” rows and “FD(ours)” rows in Table 6.1 and Table 6.2 indicate that the transformation classifier usually contributes to improved performance in our proposed approach.



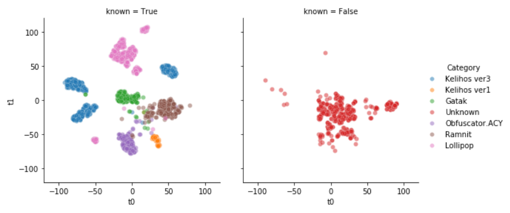
(a) OpenMax



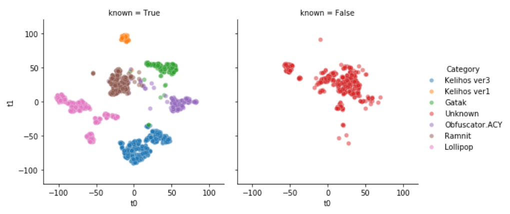
(b) No Pre-training



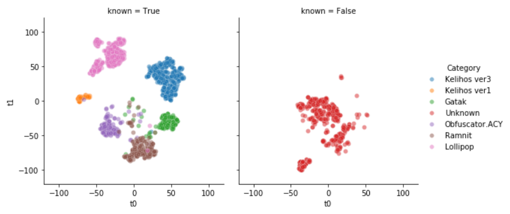
(c) RotNet



(d) Barlow Twins



(e) DTAE



(f) Feature Decoupling

Figure 6.5: The t-SNE plots of the representations of MS test samples learned by different models: (a) OpenMax; (b) triplet loss without pre-training; (c) triplet loss pre-trained with RotNet; (d) triplet loss pre-trained with Barlow Twins; (e) triplet loss pre-trained with DTAE; (f) triplet loss pre-trained with Feature Decoupling. The left subplots are the representations of the known class, and the right subplots are the representations of the unknown classes.

6.3.4 Analysis of the self-supervised models

Our experiment results indicate that the proposed feature decoupling approach benefits different loss functions on the OSR tasks. We plot the t-SNE plots at different stages to further analyze the model performances. Figure 6.3 shows the t-SNE plots of the known classes in the (unseen) test set of Fashion-MNIST after pre-training. Specifically, the known classes include “Ankle boot”, “Coat”, “Dress”, “Pullover”, “Sandal” and “Shirt”. The models are pre-trained by self-supervised learning approaches: RotNet, Barlow Twins, DTAE, and Feature Decoupling. Comparing the four approaches, we observe that RotNet fails to separate any of the six known classes, while the other three approaches manage to separate the known classes to some level. Among the other three approaches, Barlow Twins and Feature Decoupling can better cluster ”Dress” samples, whereas the representations of the ”Dress” samples learned by DTAE are more spread out and meanwhile overlap with the representations of the ”Shirt” samples. Moreover, the representations of ”Ankle boot” and ”Sandal” samples learned by Feature Decoupling are more separable than the other approaches.

Besides visually evaluating representations via t-SNE plots, we propose intra-inter ratio (IIR) to measure the representation quality learned by different self-supervised pre-training approaches. For class k, we define the intra class spread as the average distance of instances from its centroid:

$$intra_k = \frac{1}{N_k}\sum_{i=1}^{N_k} d(\mu_k, z_i), \tag{6.7}$$

where N_k is the number of samples in class k and $d(.,.)$ is a distance function. Meanwhile, we measure the inter separation of the class k as the distance of the its centroid μ_k to its nearest centroid of other classes:

$$inter_k = \min_{i, i \neq k} d(\mu_k, \mu_i) \tag{6.8}$$

Moreover, the intra-inter ratio of class k can be then defined as $IIR_k = intra_k/inter_k$, which combines both intra-spread and inter-separation for the representation quality measurement. Here, we use the average IIR over all the K known classes to further measure the representation quality:

$$IIR = \frac{1}{K} \sum_{k=1}^{K} \frac{intra_k}{inter_k} \tag{6.9}$$

IIR is similar to the feature space density proposed by Roth et al. [75]. One difference is that IIR calculates the average ratio of all classes instead of the ratio of the average intra-distance and inter-distance. That is, IIR focuses on the representation quality of each class before considering the overall quality. Also, the inter-distance in IIR is calculated with respect to the nearest centroid, while in feature space density, it is an average of all pairs of centroids. That is, inter-distance in IIR is designed to characterize the ”near miss” centroid that is most likely to cause misclassification.

A lower IIR score indicates lower intra-spread and/or higher inter-separation, which characterizes better representation quality. Figure 6.4 shows the IIR of different datasets after the pre-training stage. We observe that our proposed Feature Decoupling (FD) outperforms the other pre-training approaches for the image and malware datasets. Note that self-supervised pre-training does not use class labels, but FD can yield better representations in the t-SNE plots and lower IIR in the test set.

6.3.5 Analysis of the fine-tuned models

Besides the self-supervised learning stage, we also visualize the difference between learned representations after the fine-tuning stage under the supervised OSR scenario. Figure 6.5 shows the representations of known and unknown MS malware samples learned by different methods after the fine-tuning stage. In these experiments, we consider "Kelihos ver3", "Kelihos ver1", "Gatak", "Obfuscator.ACY", "Ramnit" and "Lollipop" as the known classes, and the samples of the remaining three classes "Vundo", "Simda", "Tracur" together are treated as the unknown class not a participant in the training process. Figure 6.5a and Figure 6.5b do not involve the pre-training stage. The model used in Figure 6.5a is trained by OpenMax, and Figure 6.5b is trained by triplet loss directly. For comparison, the models in Figure 6.5c - Figure 6.5f are pre-trained by different self-supervision approaches and fine-tuned by triplet loss. From the representations of the unknown classes in the left subplots, we observe that OpenMax, triplet loss without pre-training, and Feature Decoupling perform better in the intra-class spread. At the same time, the models pre-trained by RotNet, Barlow Twins, and DTAE tend to spread one class into several clusters, such as "Kelihos ver3" and "Lollipop". Furthermore, comparing the representations of the unknown samples in the right subplots, the representations of the unknown class learned by models in the pre-training stage are more concentrated near the origin and tend to achieve better intra-class spread. Comparing the left and right subplots, we observe that compared with other approaches, the representations of the known classes have less overlap with those of the unknown class in the Feature Decoupling approach.

Besides the triplet loss, we plot the IIR of the models fine-tuned by cross-

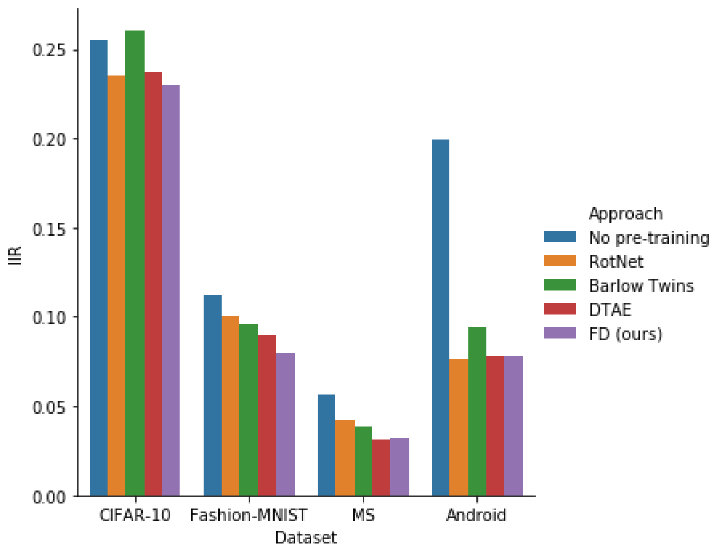


Figure 6.6: IIR after the fine-tuning stage.

entropy loss in Figure 6.6. Self-supervised pre-training benefits IIR in most cases, except for Barlow Twins in the CIFAR-10 dataset. Consistent with the IIR after the pre-training stage, the model pre-trained with Feature Decoupling benefits IIR in most cases. Furthermore, to determine if IIR can help explain OSR performance, we plot the overall F1 scores in Table 6.2 against IIR in Figure 6.7. We observe that F1 scores and IIR are highly correlated, where the Pearson correlation coefficient is -0.88. The strong correlation indicates that improvement in IIR can help explain enhancement in overall F1. Hence, self-supervised methods (such as FD) that can improve IIR can increase OSR performance.

Figure 6.8 shows the distributions of the outlier scores for the known and unknown classes in the MC dataset. Figure 6.8a and Figure 6.8b show the outlier scores distributions of the models using cross-entropy with and without Feature Decoupling pre-training, respectively. Figure 6.8c and Figure 6.8d show the outlier scores distributions of the models using triplet loss

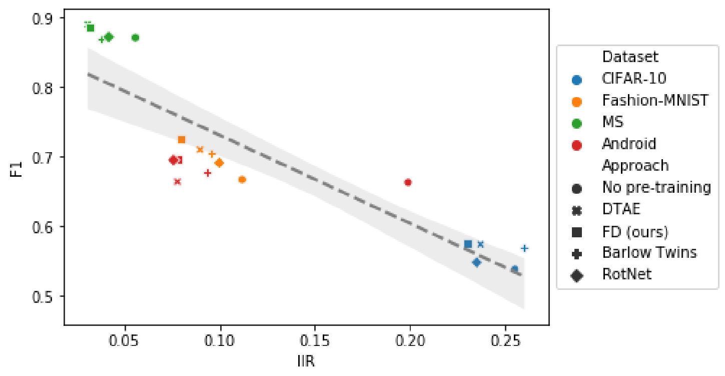
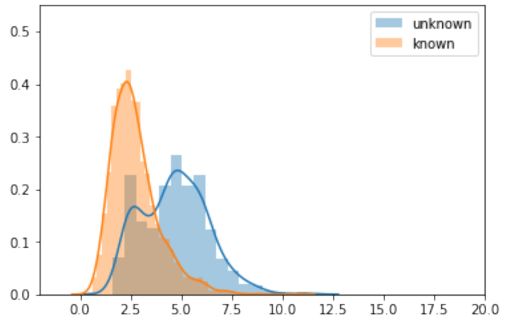


Figure 6.7: F1 against IIR after the fine-tuning stage.

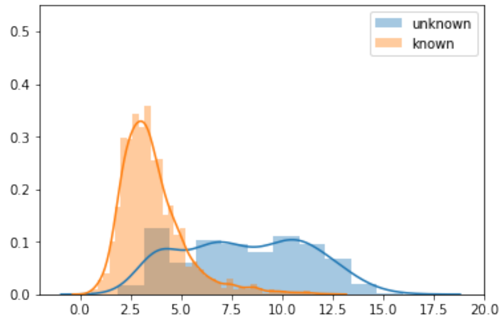
with and without Feature Decoupling pre-training, respectively. We observe that for both loss functions, the pre-training stage significantly increases outlier scores for the unknown class. Meanwhile, the outlier scores of the known classes are slightly increased. This effect pushes the outlier scores of the unknown class further away from the known classes and results in less overlap. The less overlap leads to higher accuracy of recognizing the unknown class.

6.3.6 Experiments on unsupervised OSR

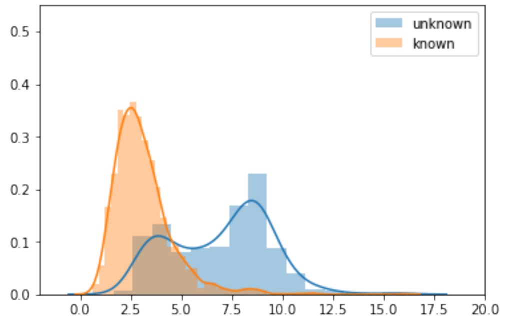
We evaluate the unsupervised scenario discussed in Section 6.2.4 on the Fashion-MNIST and MS datasets. Although both datasets contain class labels, we only use the labels to create the open-set datasets and calculate model performance metrics. We perform K-Means (K=6) on the representations learned by the self-supervised models to find the potential class centroids. Similar to the supervised scenario, we simulate an open-set scenario by randomly picking six classes as the known classes. Also, we simulate three groups of such open sets and experiment with each group with three runs, resulting 9 runs. Then, we calculate the average results of these 9 runs when evaluating the model performances. We compare our feature de-



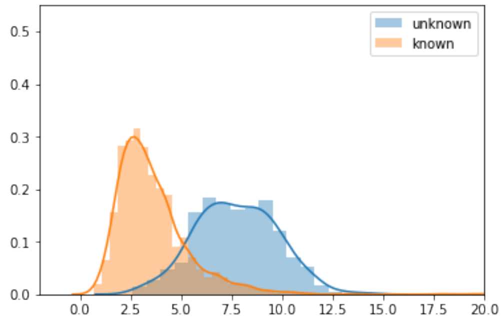
(a) Without FD (ce)



(b) With FD (ce)



(c) Without FD (triplet)



(d) With FD (triplet)

Figure 6.8: The distributions of outlier scores for the known and unknown classes of the MS dataset with and without our proposed Feature Decoupling (FD) pre-training process.

coupling (FD) approach with other self-supervised pre-training approaches, RotNet, Barlow Twins, and DTAE, and report the ROC AUC scores under 100% and 10% False Positive Rate (FPR) in Table 6.3. Our feature decoupling approach outperforms the other self-supervised learning approaches in both image and malware datasets. Though we expect AUC in the unsupervised OSR scenario (Table 6.3) to be lower than AUC in the supervised OSR scenario (Table 6.1), for feature decoupling (FD), the difference might not be huge. For Fashion-MINST, AUC with 100% FPR in unsupervised OSR is 0.655, compared to 0.771 (ce) or 0.758 (triplet) in supervised OSR. This provides additional evidence (beyond Sec. 6.3.4) that the features learned

Table 6.3: The average ROC AUC scores of 9 runs at 100% and 10% FPR of a group of 4 methods (RotNet, Barlow Twins, DTAE and Feature Decoupling (FD)) for the unsupervised OSR scenario. The values in bold are the highest values.

	Fashion-MNIST		MS	
FPR	100%	10%	100%	10%
RotNet	$0.519_{\pm 0.089}$	$0.005_{\pm 0.002}$	$0.587_{\pm 0.063}$	$0.008_{\pm 0.002}$
Barlow Twins	$0.463_{\pm 0.059}$	$0.004_{\pm 0.002}$	$0.537_{\pm 0.120}$	$0.007_{\pm 0.002}$
DTAE	$0.639_{\pm 0.083}$	$0.032_{\pm 0.006}$	$0.639_{\pm 0.083}$	$0.010_{\pm 0.001}$
FD (ours)	$\mathbf{0.655}_{\pm 0.053}$	$\mathbf{0.034}_{\pm 0.006}$	$\mathbf{0.686}_{\pm 0.092}$	$\mathbf{0.012}_{\pm 0.005}$

via feature decoupling could be effective for OSR.

6.4 Conclusion

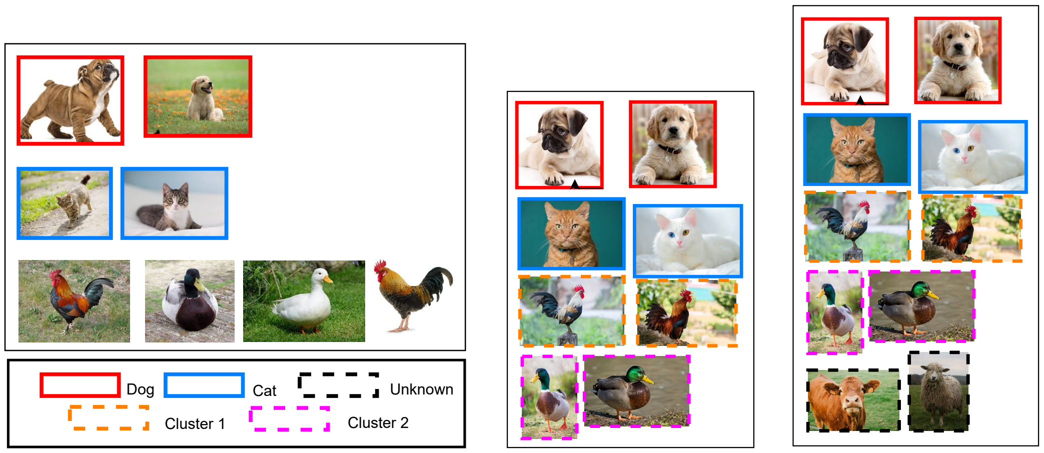
We use a two-stage learning approach for the OSR problems. We propose a self-supervised feature decoupling method to split the learned representation into the content and transformation parts in the first stage. In the second stage, we fine-tune the content features from the first stage with class labels. Furthermore, we consider an unsupervised OSR scenario, where we cluster the content features to find the potential classes in the second stage. We introduce intra-inter ratio (IIR) to evaluate the learned content representations. The results indicate that our feature decoupling method outperforms the other self-supervised learning methods in supervised and unsupervised OSR scenarios with image and malware datasets. Our analyses indicate that IIR is correlated with and can explain OSR performance.

Chapter 7

Novel Category Discovery in Open Set Recognition

7.1 Introduction

Machine learning models have achieved significant advances in various tasks in recent years. Most of these models are developed under a closed-world assumption and rely on a huge amount of data with human annotations. The real world is an open set, and humans can determine whether images belong to the same category or not. However, such an open-set setting brings new challenges for machine learning models. First, it is cost-inhibitive to keep manually annotating the emerging new categories. Second, it is unlikely to collect samples exhausting all the classes. In the open-set setting, an ideal machine learning model should automatically discover new categories in the training set without having access to their labels, called novel category discovery (NCD) [36]. Meanwhile, the model should recognize the unknown



(a) Training samples (b) Conventional NCD (c) NCD under open-set scenario

Figure 7.1: The differences between conventional NCD and NCD under open-set scenario.

classes absent from the training set, which is referred as Open Set Recognition (OSR) [4].

In this chapter, we focus on automatically discovering novel categories in a more realistic open-set scenario. In the open-set setting, we have labeled and unlabeled samples available for training. Meantime we have unknown samples that are not available in the training process. Our proposed approach has three objectives: classifying the existing categories from the labeled samples, clustering the novel categories from the unlabeled samples, and recognizing the unknown classes absent from the training set. As shown in the example in Figure 7.1, we have labeled "dog" and "cat" samples and unlabeled "chicken" and "duck" samples for training. Conventional NCD methods classify the "dog" and "cat" samples, meanwhile cluster the "chicken" and "duck" samples, as shown in Figure 7.1b. For NCD under the open-set scenario in Figure 7.1c, the ideal system not only recognizes the existing ("dog", "cat") and novel categories ("chicken", "duck"), but also rejects the classes ("sheep", "cow") that absent from the training samples as unknown.

Specifically, we introduce a one-step solution for NCD under the open-set scenario and name this solution general inter-intra (GII) loss. Mehadi and Chan [37] propose inter-intra (ii) loss for OSR with labeled training samples. Ii loss maximizes the inter-class distances and minimizes the intra-class distances in the representation space to achieve inter-class separation and intra-class compactness. We generalize this idea to unlabeled samples in our work. The proposed GII consists of three components: intra-class loss for existing categories, intra-cluster loss for novel categories, and inter-category loss for merged categories. We calculate their class centroids in representation space for existing categories and minimize the intra-class distance. For novel categories, we first estimate the centroids of the novel categories and cluster assignments via k-means, then we minimize the intra-cluster distance in the representation space. The assumption is that novel categories are totally disjoint with the existing ones, so intra-category loss is designed to maximize the distance between any two categories.

Our contribution includes: first, we propose a generic, one-step solution for NCD under an open-set scenario. Second, to the best of our knowledge, we are the first to extend NCD to an open-set setting. Third, we experiment with the proposed approach with image and graph datasets, and the results indicate that our proposed approach is more effective than other approaches for NCD and OSR.

We organize this chapter as follows. Section 7.2 presents our one-step solution for NCD under open-set scenario. In Section 7.3, we evaluate our proposed approach through experiments on different types of datasets and also compare the experimental results with other approaches.

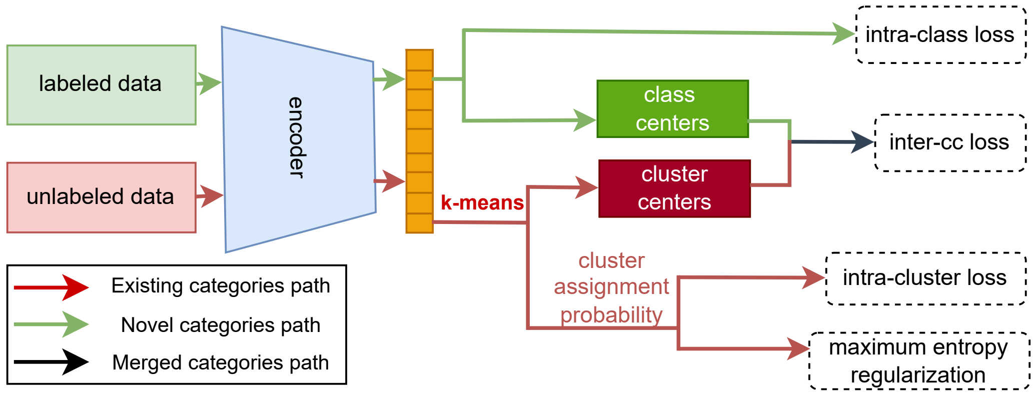


Figure 7.2: Illustration of GII architecture for NCD.

7.2 Approach

This section includes details on the General Intra-Inter (GII) loss for the NCD under an open-set scenario. Hassen and Chan [41] propose ii loss for OSR, where the training samples are all labeled data. Ii loss learns the representations that encourage intra-class compactness and inter-class separability. In this work, we generalize a similar motivation to the unlabeled data for NCD. As shown in Figure 7.2, our proposed network architecture consists of three paths: the existing categories path decreases the intra-class spread for the labeled samples; the novel categories path uses k-means to estimate the cluster centroids as well as cluster assignments for the unlabeled samples, then assist intra-cluster compactness; merged categories path deals with both existing categories and novel categories by increasing inter-category separation.

7.2.1 Learning Representations of Existing and Novel Categories

Suppose we have a labeled collection of instances $D^l = \{(x_i^l, y_i^l)\}_{i=1}^{N^l}$, where $y_i^l \in \{1, \ldots, C^l\}$ is the ground-truth class labels for the labeled samples, and

N^l is the number of labeled samples. In addition, we have an unlabelled collection of instances $D^u = \{x_i^u\}_{i=1}^{N^u}$, where N^u is the number of unlabelled samples. Following a common assumption in other works [36, 35], we assume that the novel categories are disjoint with the existing ones, i.e., $D^l \cap D^u = \emptyset$, also the number of novel categories C^u is known.

Our goal is to model a representation space that separates the existing categories in D^l and the novel categories in D^u. Through such representation space, we can identify if a test instance belongs to one of the existing categories, one of the novel categories, or the unknown class. We propose an end-to-end framework to learn the representations, which provides a one-step solution for NCD under the open-set scenario. The end-to-end training of the framework consists of three components: intra-class loss for the existing categories, intra-cluster loss for the novel categories, and inter-category loss for the merged categories. The existing categories are the classes of the labeled samples. The novel categories are the clusters of the unlabeled samples, and the merged categories are the combinations of these classes and clusters.

7.2.1.1 Intra-class loss for existing categories

The intra-class component deals with the intra-spread for the labeled samples. One can expect the network to capture some informative knowledge for the existing categories through the training process, which not only helps classify labeled samples but also is beneficial to transfer the basic feature for clustering unlabeled samples.

Given a labeled sample x_i^l, we use a network-based trainable decoder $f(\cdot)$ to extract its representation vector z_i^l. Thus, for existing category (or class) j, we find its centroid in the representation space as:

$$\mu_j^l = \frac{1}{N_j^l} \sum_{i=1}^{N_j^l} z_i^l, \tag{7.1}$$

where N_j^l denotes the number of samples in the existing category j. Then, we measure the intra-class spread as the average distance of labeled instances from their class means:

$$\text{intra-class}_j = \frac{1}{N_j^l} \sum_{i=1}^{N_j^l} \|\mu_j^l - z_i^l\|_2^2. \tag{7.2}$$

To improve the intra-class compactness, we minimize the largest intra-class spread among the existing categories.

$$\mathcal{L}_{\text{intra-class}} = \max_{1 \leq j \leq C^l} \text{intra-class}_j \tag{7.3}$$

7.2.1.2 Intra-cluster loss for novel categories

There are several differences comparing intra-cluster spread with intra-class spread. First, intra-class spread relies on labels to find class centroids. In the intra-cluster spread, we only have unlabeled samples. Thus, we use k-means to estimate the representation cluster centroids as the centers of novel categories $\tilde{\mu}^u$. Second, we are uncertain which specific centroid is for an unlabeled sample. Thus, we calculate the soft assignment of sample x_i^u based on the distance of its representation z_i^u to the estimated centroids. Since unlabeled samples do not belong to known classes, these samples do not have a soft assignment to known classes. To calculate the soft assignment (probability), we use the softmax of the negative distance of z_i^u from all the estimated centroids. Hence, the probability of sample x_i^u belongs to novel category (or cluster) k is given by:

$$p_{ik} = P(y_i^u = k|x_i^u) = \frac{e^{-\|\tilde{\mu}_k^u - z_i^u\|_2^2}}{\sum_{t=1}^{C^u} e^{-\|\tilde{\mu}_t^u - z_i^u\|_2^2}}, \tag{7.4}$$

where $\tilde{\mu}_k^u$ is the estimated centroid for novel category k. Similar to the intra-class spread, we measure the intra-cluster spread as the weighted average distance of unlabeled instances from their soft assignments. Suppose we have N_u unlabeled samples, the intra-cluster spread of novel category k is calculated as:

$$\text{intra-cluster}_k = \frac{\sum_{i=1}^{N^u} p_{ik} \|\tilde{\mu}_k^u - z_i^u\|_2^2}{\sum_{i=1}^{N^u} p_{ik}}. \tag{7.5}$$

Then, we minimize the largest intra-cluster spread among the novel categories to achieve intra-cluster compactness. The differences between the intra-cluster spread in Equation 7.5 with the intra-class spread in Equation 7.2 are the estimated cluster centroid $\tilde{\mu}_k^u$ and the soft assignment p_{ik}.

$$\mathcal{L}_{\text{intra-cluster}} = \max_{1 \le k \le C^u} \text{intra-cluster}_k \tag{7.6}$$

It can be seen that intra-cluster loss sharpens the distribution of soft assignments through the training process.

The cluster centroids are initialized and updated by k-means. To reduce the training time, we use a scheduling function for the k-means. Intuitively, we want to update the centroids more frequently at the beginning of the training. Close to the end of the training, when the network has learned informative knowledge from the labeled samples, and the clusters of the unlabeled samples have been formed for the novel categories, we perform k-means less frequently for the centroids updates.

Finally, to avoid a trivial solution of assigning all unlabeled samples to the same class, we regularize the model with maximum entropy regularization (MER). Specifically, we use the probability p_{ik} calculated from Equation 6.5 as the probability of an unlabeled sample x_i^u being assigned to novel category k. MER maximizes the entropy of the output probability distribution:

$$\mathcal{R} = -H(p) = \frac{1}{N^u} \sum_{i=1}^{N^u} \sum_{k=1}^{C^u} p_{ik} \log p_{ik}. \tag{7.7}$$

MER has been used in pseudo-labeling based semi-supervised learning [1, 9] and deep clustering methods [22].

7.2.1.3 Inter-category loss for merged categories

The above two components shorten the distance between representations of the same categories to ensure intra-class and intra-cluster compactness. To distribute the representations of different categories to different subspaces, we further measure the inter-category separation as the distance between the two closest category centroids. Let μ_c be the centroid of category c, where $c \in \{1, ..., C^l\} \bigcup \{1, ..., C^u\}$. The inter-category separation for category m is defined as:

$$\text{inter-category}_m = \min_{1 \leq i \leq (C^l + C^u), k \neq i} \|\mu_m - \mu_i\|_2^2. \tag{7.8}$$

To improve the intra-category separability, we maximize the inter-category separation in the inter-category loss:

$$\mathcal{L}_{\text{inter-category}} = - \min_{1 \leq m \leq (C^l + C^u)} \text{inter-category}_m. \tag{7.9}$$

The objective function in GII combines three components, and the overall training loss of our end-to-end framework can then be written as:

$$\mathcal{L} = \mathcal{L}_{\text{intra-class}} + \lambda_1 \mathcal{L}_{\text{intra-cluster}} + \lambda_2 \mathcal{L}_{\text{inter-category}} + \lambda_3 \mathcal{R}, \tag{7.10}$$

where λ_1, λ_2, and λ_3 are regularization parameters set to 1 in all our experiments.

Algorithm 3 Training to minimize GII loss

```
    Input: Labeled samples and their labels $\{(x^l, y^l)\}$. Unlabeled samples $\{x^u\}$.
    Output: Encoder $f(\cdot)$. Cluster assignment $\tilde{y}^u$.
            Representation centroids of all categories $\{\mu^l, \mu^u\}$.

 1: Initialize $f(\cdot)$
 2: Initialize $\{\mu^u\}$ via k-means($x^u$)
 3: for number of iterations do
 4:     if k-means scheduler is on then
 5:         Update cluster centroids $\{\tilde{\mu}^u\} \leftarrow$ k-means($x^u$)
 6:     Sample a mini-batch from the training set.
 7:     Extract the representations $z^l \leftarrow f(x^l), z^u \leftarrow f(x^u)$
 8:     Calculate class centroids $\{\mu^l\} \leftarrow$ mean($z^l$)                ▷ Eq. 7.1
 9:     Calculate intra-class loss $\mathcal{L}_{\text{intra-class}}(\mu^l, z^l)$             ▷ Eq. 7.3
10:     Get the prediction probability $p$ for unlabeled data         ▷ Eq.6.5
11:     Calculate intra-cluster loss $\mathcal{L}_{\text{intra-cluster}}(\tilde{\mu}^u, z^u)$        ▷ Eq. 7.6
12:     Calculate inter-category loss $\mathcal{L}_{\text{inter-category}}(\tilde{\mu}^u, \tilde{\mu}^l)$     ▷ Eq. 7.9
13:     Calculate MER $\mathcal{R} \leftarrow -H(p)$                          ▷ Eq. 7.7
14:     Compute the joint loss $\mathcal{L}(\mathcal{L}_{\text{intra-class}}, \mathcal{L}_{\text{intra-cluster}}, \mathcal{L}_{\text{inter-category}}, \mathcal{R})$   ▷ Eq. 7.10
15:     Update $f(\cdot)$ via back-propagation
16: Update cluster centroids and assignments $\{\mu^u\}, \tilde{y}^u \leftarrow$ k-mean($z^u$).
17: return $f(\cdot), \{\mu^l, \mu^u\}, \tilde{y}^u$
```

In Algorithm 3, we summarize the overall training process for GII loss. After Initializing the networks in Line 1 and estimating the cluster centroids in Line 2, the network is trained using mini-batch stochastic gradient descent with back-propagation. During each iteration, we first check if it is time to update the cluster centroids based on the pre-defined scheduling function.

Then we update the cluster centroids via k-means if the schedular is on (Line 4- Line 5). After extracting the representations of labeled and unlabeled samples, we calculate class centroids for labeled samples in Line 8. Lines 9 - 12 calculate inter-class, intra-cluster, and inter-category loss separately. Then, to avoid a trivial solution, we calculate MER as in Line 13. Line 14 computes the joint loss of the three components. Then we update the network parameters to minimize the loss value via back-propagation in Line 15. Finally, after the training process, we get the cluster centroids and cluster assignments via k-means.

7.2.2 Open Set Recognition

After training the encoder and obtaining the category centroids, we utilize the distances between the representations and the centroids for the NCD and OSR tasks.

Since different categories have different spreads in the representation space, we first normalize the distances between the representations to their category centroids. Specifically, or category k (k can be either existing category or novel category), we calculate the mean m_k and standard deviation s_k of the category spread as the distance from training sample i in category k to their centroid μ_k.

$$\begin{aligned} m_k =& \frac{1}{N_k}\sum_{i=1}^{N_k} D(z_i, \mu_k) \\ s_k =& \sqrt{\frac{\sum_{i=1}^{N_k}(D(z_i, \mu_k) - m_k)^2}{N_k}}, \end{aligned} \tag{7.11}$$

where N_k is the number of training samples in category k. Given the representation of a test sample z, we normalize its distance to category centroid

μ_k as:

$$d_k^{\text{norm}} = \frac{\|D(\mu_k, z) - m_k\|}{s_k}. \tag{7.12}$$

Then, for sample x, we calculate its probability belonging to category k as:

$$P(y = k|x) = \frac{e^{-d_k^{norm}}}{\sum_{k=1}^{C^l+C^u} e^{-d_k^{norm}}} \tag{7.13}$$

Furthermore, the test sample is classified as the category with the highest probability if the probability is above the threshold t. Otherwise, the test sample is identified as the unknown class.

$$\hat{y} = \begin{cases} \text{unknown}, & \text{if } \max_{1 \le k \le (C^l+C^u)} P(y = k|x) < t \\ \underset{1 \le k \le (C^l+C^u)}{\text{argmax}} \; P(y = k|x), & \text{otherwise} \end{cases} \tag{7.14}$$

7.3 Experimental Evaluation

In this section, our proposed GII is systemically evaluated on image and graph datasets.

MNIST [74] contains 70,000 handwritten digits from 0 to 9. Each example in the MNIST dataset is a 28x28 grayscale image.

Fashion-MNIST [74] is associated with 10 classes of clothing images. It contains 60,000 training and 10,000 testing examples. In the Fashion-MNIST dataset, each example is a 28x28 grayscale image.

Microsoft Challenge (MS) [74] contains disassembled malware samples from 9 families:“Ramnit”, “Lollipop”, “Kelihos ver3”, “Vundo”, “Simda”, “Tracur”, “Kelihos ver1”, “Obfuscator.ACY ” and “Gatak”. We use 10260 samples that can be correctly parsed and then extracted their FCGs as in [39] for the experiment.

Android Genome (AG) consists of 1,113 benign android apps and 1,200 malicious android apps. Our colleague provides the benign samples, and the malicious samples are from [108]. We select nine families with a relatively larger size for the experiment to be fairly split into the training set and the test set. The nine families contain 986 samples in total. We first use [23] to extract the function instructions and then generated the FCGs as in [39].

7.3.1 Implementation details

To simulate an open-set scenario, we randomly select six classes from the datasets as existing categories. Moreover, we randomly select another two classes from the datasets as novel categories by removing their labels. These eight classes participate in the training, while the rest are considered unknowns that only exist in the test set.

As shown in Figure 7.2, labeled and unlabeled data share the same encoder. For the MNIST and Fashion-MNIST datasets, the padded input layer of the encoder 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 layer. We use two fully connected non-linear layers with 256 and 128 hidden units after the convolutional component. Then we have an eight-dimensional representation layer after the encoder. We use the Relu activation function for all the non-linear layers and set the Dropout rate as 0.2 for the fully connected layers. We use Adam optimizer with a learning rate of 0.001. We use a contamination ratio of 0.001 for the unknown class threshold selection. We sort the output probability of training data in ascending order and pick the 0.1 percentile of the probability as the threshold.

For the FCG datasets (MS and Android), the padded input layer is in the size of (67, 67). The padded input layer is then flowed by two non-linear convolutional layers with 32 and 64 nodes. We apply the max-polling layers with kernel size (3, 3) and strides (2, 2). We also add batch normalization after each convolutional layer to complete the convolutional block. After the convolutional block, we only use one fully connected non-linear layer with 256 hidden units for the graph dataset. Next, we add an eight-dimensional representation layer after the encoder same as the Fashion-MNIST dataset. We use the Relu activation function and set the Dropout rate as 0.2. We use Adam as the optimizer with a learning rate of 0.001. Finally, we use a contamination ratio of 0.01 for the unknown class threshold selection.

Moreover, as mentioned in section 7.2.1.2, we use a scheduling function for the k-means updates in the NCD process. In the experiments, we apply k-means every ten iterations in the first 5000 iterations, then reduce the frequency to every 100 iterations in the rest of the training process.

7.3.2 Comparison methods

We compare the proposed with ii loss without sharpening on the unlabeled samples (No sharpening) , cluster loss, and supervised OSR. For a fair comparison with "No sharpening", we first pre-train the encoder with labeled samples using ii loss [41]. After obtaining the representations of the unlabeled samples, we find the novel cluster centroids and assignments via k-means directly in the representation space without further sharpening. Finally, we apply the same OSR process as described in section 5.2.5.

Liu and Tuytelaars [64] propose cluster loss to sharpen the distribution of unlabeled samples through the clustering process. Specifically, they construct an auxiliary target distribution as a sharper version of the distribution of unlabeled samples and minimize the KL-divergence loss between the target distribution and actual distribution. While our proposed intra-cluster loss in section 7.2.1.2 can be seen as a sharpening process for the unlabeled samples as well, we compare our proposed intra-cluster loss with cluster loss by substituting the inter-cluster loss term with cluster loss in our overall loss function in Equation 7.10. Moreover, as the cluster loss measures the KL-divergence between two distributions, which is in a different scale with other terms (intra-class and inter-category), we set λ_1 differently for different datasets.

In addition, we experiment on fully supervised OSR and use the results as the upper bounds of NCD and OSR performances. In the supervised OSR experiments, we apply ii loss on eight labeled categories in the training process. The remaining categories are considered as the unknown class and only participants in the testing phase.

7.3.3 Evaluation Criteria

As mentioned above, we simulate an open-set scenario for all the datasets. Moreover, we randomly select two classes in the training set as novel categories and remove their class labels. We simulate three open-set groups for each dataset and then repeat each group 10 runs, so each dataset has results for 30 runs. We calculate the average results of the 30 runs for performance evaluation.

We calculate the accuracy (ACC) scores under different types of categories: existing categories ($\mathrm{ACC_E}$), novel categories ($\mathrm{ACC_N}$) and the unknown category ($\mathrm{ACC_U}$). Specifically, we evaluate the classification accuracy of existing categories and the recognition accuracy of the unknown category.

Moreover, we evaluate the model performance on novel categories with clustering accuracy. Clustering accuracy is widely used in NCD problems. To find the optimal match between the class labels and the cluster labels, the ACC of novel categories is defined as:

$$\mathrm{ACC_N} = \max_{\mathrm{perm} \in P} \frac{1}{N} \sum_{i=1}^{N} \delta(\mathrm{perm}(\hat{y}_i) = y_i), \tag{7.15}$$

where N is the total number of unlabeled samples; δ is the Kronecker delta response; $\hat{y}_i$ denotes the predicted cluster label; $\mathrm{perm}(\cdot)$ is the permutation operation and P is the set of all permutations of the class assignments in the test set. The score ranges between 0 and 1, and a higher value means better clustering performance. The Hungarian algorithm is commonly used to optimize the permutations for faster computation.

To further evaluate our approach's performance on OSR, we measure the AUC scores under 100% and 10% False Positive Rate (FPR). While the AUC score under 100% FPR is commonly used in model performance measurements, the AUC score under 10% FPR is more meaningful for malware detection applications.

Table 7.1: The average ACC scores of 30 runs. The upper bounds results are trained with fully supervised learning, and the values in boldface are the highest in each column.

Image Dataset	MNIST					Fashion-MNIST				
	$\mathrm{ACC_E}$	$\mathrm{ACC_N}$	$\mathrm{ACC_{E+N}}$	$\mathrm{ACC_U}$	$\mathrm{ACC_{E+N+U}}$	$\mathrm{ACC_E}$	$\mathrm{ACC_N}$	$\mathrm{ACC_{E+N}}$	$\mathrm{ACC_U}$	$\mathrm{ACC_{E+N+U}}$
No sharpening	$0.733_{\pm 0.078}$	$0.800_{\pm 0.091}$	$0.697_{\pm 0.078}$	$0.767_{\pm 0.015}$	$0.615_{\pm 0.060}$	$0.598_{\pm 0.068}$	$0.668_{\pm 0.089}$	$0.539_{\pm 0.098}$	$0.786_{\pm 0.008}$	$0.468_{\pm 0.079}$
Cluster loss	$0.752_{\pm 0.161}$	$0.625_{\pm 0.125}$	$0.687_{\pm 0.166}$	$0.751_{\pm 0.031}$	$0.624_{\pm 0.127}$	$0.820_{\pm 0.062}$	$0.608_{\pm 0.104}$	$0.757_{\pm 0.052}$	$0.698_{\pm 0.049}$	$0.628_{\pm 0.042}$
GII (ours)	$\mathbf{0.936}_{\pm 0.08}$	$\mathbf{0.854}_{\pm 0.088}$	$\mathbf{0.909}_{\pm 0.089}$	$\mathbf{0.817}_{\pm 0.070}$	$\mathbf{0.810}_{\pm 0.069}$	$\mathbf{0.875}_{\pm 0.047}$	$\mathbf{0.808}_{\pm 0.084}$	$\mathbf{0.847}_{\pm 0.051}$	$\mathbf{0.797}_{\pm 0.003}$	$\mathbf{0.687}_{\pm 0.034}$
Upper bound (supervised)	$0.983_{\pm 0.001}$	$0.977_{\pm 0.004}$	$0.981_{\pm 0.001}$	$0.937_{\pm 0.012}$	$0.935_{\pm 0.012}$	$0.896_{\pm 0.018}$	$0.967_{\pm 0.005}$	$0.914_{\pm 0.014}$	$0.822_{\pm 0.011}$	$0.770_{\pm 0.016}$
Malware Dataset	MS					AG				
	$\mathrm{ACC_E}$	$\mathrm{ACC_N}$	$\mathrm{ACC_{E+N}}$	$\mathrm{ACC_U}$	$\mathrm{ACC_{E+N+U}}$	$\mathrm{ACC_E}$	$\mathrm{ACC_N}$	$\mathrm{ACC_{E+N}}$	$\mathrm{ACC_U}$	$\mathrm{ACC_{E+N+U}}$
No sharpening	$0.732_{\pm 0.131}$	$0.625_{\pm 0.180}$	$0.717_{\pm 0.132}$	$0.763_{\pm 0.112}$	$0.653_{\pm 0.166}$	$0.680_{\pm 0.167}$	$0.708_{\pm 0.140}$	$0.602_{\pm 0.176}$	$0.798_{\pm 0.027}$	$0.564_{\pm 0.193}$
Cluster loss	$0.880_{\pm 0.117}$	$0.602_{\pm 0.183}$	$0.818_{\pm 0.106}$	$0.758_{\pm 0.096}$	$0.742_{\pm 0.094}$	$0.779_{\pm 0.146}$	$0.601_{\pm 0.177}$	$0.734_{\pm 0.120}$	$0.773_{\pm 0.063}$	$0.684_{\pm 0.118}$
GII (ours)	$\mathbf{0.942}_{\pm 0.026}$	$\mathbf{0.630}_{\pm 0.143}$	$\mathbf{0.895}_{\pm 0.054}$	$\mathbf{0.834}_{\pm 0.071}$	$\mathbf{0.811}_{\pm 0.078}$	$\mathbf{0.944}_{\pm 0.013}$	$\mathbf{0.714}_{\pm 0.080}$	$\mathbf{0.906}_{\pm 0.020}$	$\mathbf{0.831}_{\pm 0.048}$	$\mathbf{0.820}_{\pm 0.034}$
Upper bound (supervised)	$0.960_{\pm 0.016}$	$0.916_{\pm 0.035}$	$0.950_{\pm 0.020}$	$0.903_{\pm 0.035}$	$0.899_{\pm 0.035}$	$0.922_{\pm 0.012}$	$0.712_{\pm 0.080}$	$0.898_{\pm 0.021}$	$0.908_{\pm 0.013}$	$0.904_{\pm 0.012}$

Table 7.2: The average ROC AUC scores of 30 runs at 100% and 10% FPR. The upper bounds results are trained with fully supervised learning, and the values in boldface are the highest in each column.

	MNIST		Fashion-MNIST		MS		AG	
FPR	100%	10%	100%	10%	100%	10%	100%	10%
No sharpening	$0.439_{\pm 0.127}$	$0.004_{\pm 0.003}$	$0.418_{\pm 0.073}$	$0.003_{\pm 0.001}$	$0.528_{\pm 0.122}$	$0.007_{\pm 0.004}$	$0.293_{\pm 0.214}$	$0.000_{\pm 0.000}$
Cluster loss	$0.413_{\pm 0.231}$	$0.007_{\pm 0.009}$	$0.620_{\pm 0.084}$	$0.008_{\pm 0.003}$	$0.651_{\pm 0.271}$	$0.018_{\pm 0.015}$	$0.507_{\pm 0.283}$	$0.007_{\pm 0.015}$
GII (ours)	$\mathbf{0.829}_{\pm 0.104}$	$\mathbf{0.047}_{\pm 0.016}$	$\mathbf{0.674}_{\pm 0.040}$	$\mathbf{0.012}_{\pm 0.004}$	$\mathbf{0.858}_{\pm 0.086}$	$\mathbf{0.028}_{\pm 0.015}$	$\mathbf{0.885}_{\pm 0.090}$	$\mathbf{0.016}_{\pm 0.020}$
Upper bound (supervised)	$0.966_{\pm 0.010}$	$0.078_{\pm 0.003}$	$0.676_{\pm 0.062}$	$0.015_{\pm 0.002}$	$0.945_{\pm 0.045}$	$0.062_{\pm 0.017}$	$0.963_{\pm 0.013}$	$0.052_{\pm 0.015}$

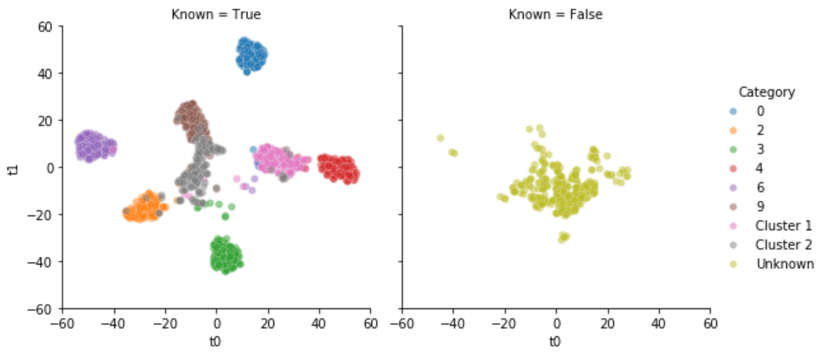
7.3.4 Experimental Results

We test our proposed method on image and malware datasets for 30 runs. Table 7.1 shows the average accuracy scores of different methods. Notably, we measure the average clustering/classification accuracy on the existing/novel set and the merged set ($\mathrm{ACC_{E+N}}$). Moreover, considering an open-set scenario, we measure the average accuracy of the unknown set, and the set contains all the existing, novel, and unknown categories ($\mathrm{ACC_{E+N+U}}$). Comparing the ACC under existing categories ($\mathrm{ACC_E}$) and novel categories ($\mathrm{ACC_N}$), we observe that our proposed GII outperforms both ii loss without sharpening and cluster loss in NCD. Also, comparing the ACC under the unknown category ($\mathrm{ACC_U}$), we observe that GII achieves the best performance in OSR. The upper bound performances are generated from supervised ii loss, where we utilize the labels of novel categories in the training set. We can see that GII has comparable performances with the supervised training in some datasets. In particular, GII obtains higher accuracy than supervised learning in the combined novel and existing categories ($\mathrm{ACC}_{\mathrm{E+N}}$) in the AG dataset.

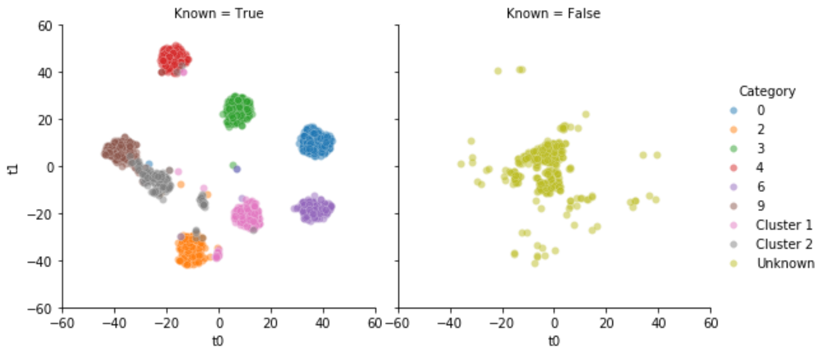
In addition to the ACC scores, we measure the AUC ROC scores under different FPR values: 100% and 10% in Table 7.2. The AUC ROC measures OSR at various threshold settings. Similar to the ACC scores, our proposed GII outperforms ii loss without sharpening and cluster loss in the AUC ROC scores. Furthermore, comparing GII with supervised learning, we observe that GII can achieve comparable OSR performance in the Fashion-MNIST dataset.

7.3.5 Analysis

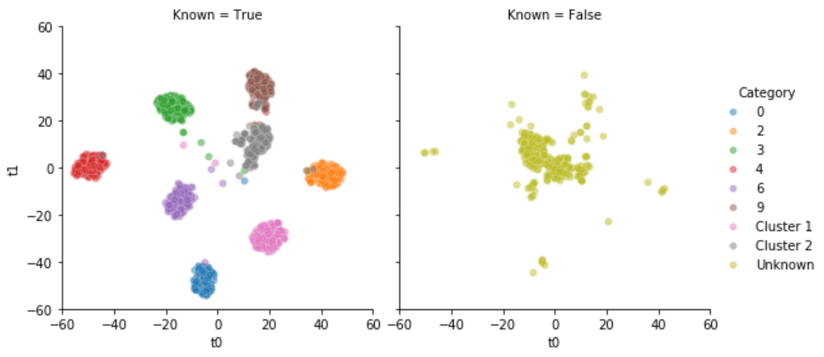
Our experiment results indicate that GII outperforms ii loss without sharpening and cluster loss in terms of performances in NCD and OSR. We plot the t-SNE plots of the representations of samples from different categories in the MNIST test set, as shown in Figure 7.3. The left subplots are the representations of the samples from existing categories ("0", "2". "3", "4", "6" and "9") and novel categories ("cluster 1" and "cluster 2"). The right subplots show the representations of samples from unknown categories, which only exist in the test set. Comparing Figure 7.3a with Figures 7.3b and 7.3c, we can see that samples from the two clusters result in more compact intra-cluster spread with cluster loss and GII. The reason is that cluster and GII sharpen the distributions of the unlabeled samples while "No sharpening" does not change the distributions of the unlabeled samples. Furthermore, it can be seen that GII forms better clusters compared with cluster loss. GII generates a more discriminative boundary for the samples in cluster 2 (grey) and the



(a) No sharpening

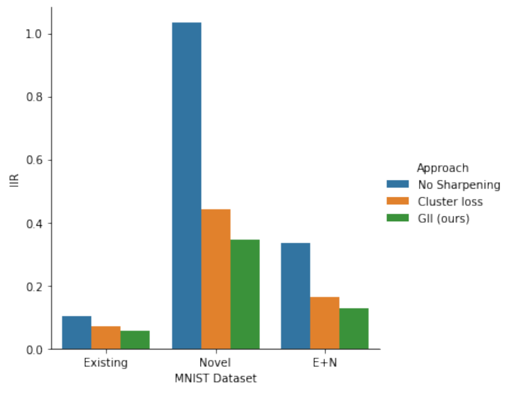


(b) Cluster loss

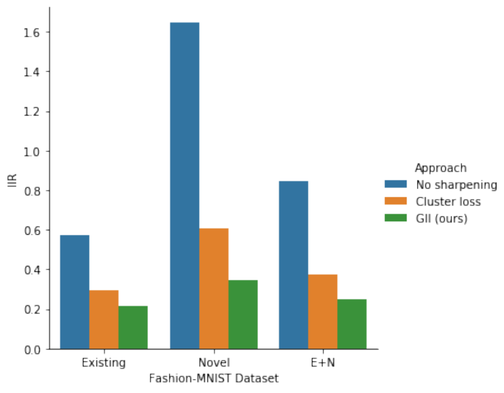


(c) GII (ours)

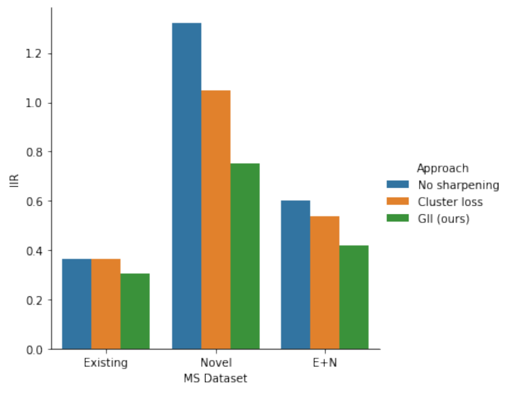
Figure 7.3: The t-SNE plots of the representations of MNIST test samples learned by different models.



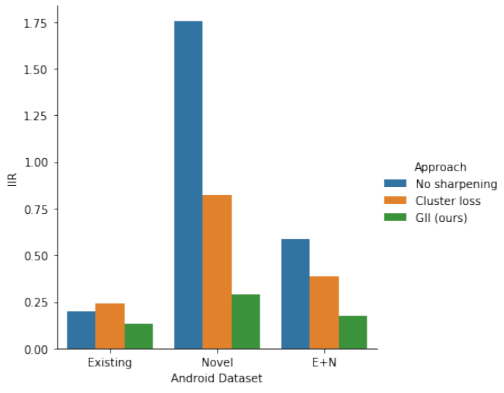
(a) MNIST



(b) Fashion-MNIST



(c) MS



(d) Android

Figure 7.4: Intra-inter ratio (IIR) of the representations in different categories

samples in class “9” (brown). The reason is that GII forms a tighter cluster for cluster 2. Thus a more accurate cluster centroid is estimated and used in the inter-category loss. Also, comparing the representations in the right subplots, we find that the representations of unknown samples learned by ii loss without sharpening and GII are more concentrated around the origin. In contrast, those learned by GII are more widespread.

Besides visually evaluating representations via t-SNE plots, we also evaluate intra-inter ratio (IIR) [49] with test samples to measure the representation quality learned by different approaches. IIR measures the representation quality by calculating the ratio between intra-category spread and inter-category separation, and a lower value means better representations. Here, we calculate inter-category separation as the average distance between any two nearest category centroids. Moreover, we calculate intra-category spread differently of different categories. We measure the intra-category spread for novel categories as the average distances between two novel category centroids. Moreover, we use the average distances between two existing category centroids as the intra-category spread of existing categories. For the combined categories (N+E), we calculate the intra-category spread as the average distance between any two combined category centroids.

When calculating the IIR, we desire a relatively small intra-category spread with a relatively large inter-category separation. Hence, a lower IIR means better representations. To find the reference point for the desired IIR score, we assume that the representations of one category are distributed in a sphere. Then, the intra-category spread represents the radius of this sphere, and the inter-category separation is the distance between its centroid and another nearest centroid. For example, if we have the centroid of category 1 as

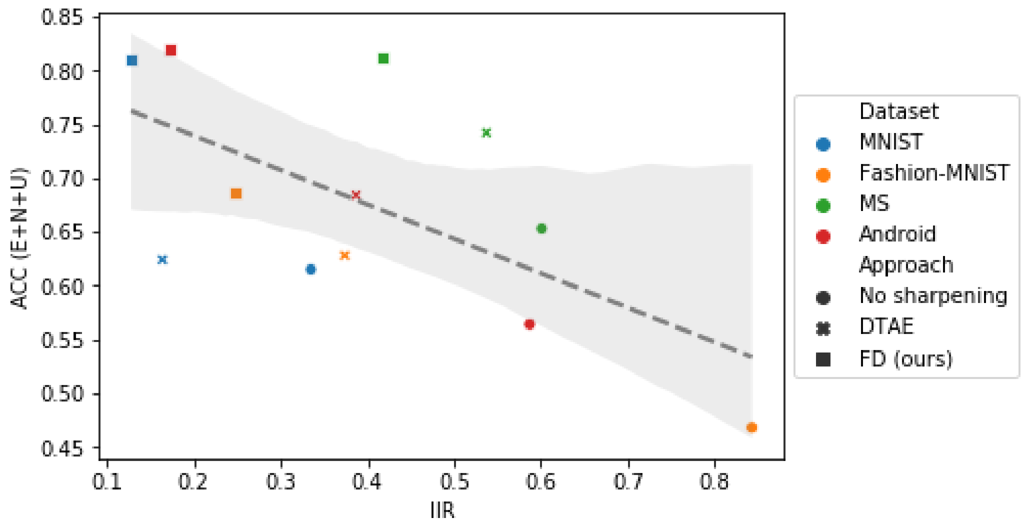


Figure 7.5: ACC against IIR in four datasets

"c1" with intra-category spread (radius) "r1". The nearest centroid to "c1" is "c2" from category 2 with intra-category spread (radius) "r2". If category 1 and category 2 are separable, the distance between "c1" and "c2" (inter-category separation, denote as "d") should be greater than "r1+r2". The IIR of these two categories is calculated as IIR $= \frac{1}{2}\frac{r1+r2}{d}$ with $d > (r2 + r2)$, which makes IIR less than 0.5. Thus, 0.5 can be used as a reference point for the desired IIR score.

Figure 7.4 shows the IIR values of different datasets. We find that cluster loss and GII achieve better IIR in novel categories for both image and graph datasets, consistent with the earlier discussion in the t-SNE plot. Also, all three approaches have comparable performances in the existing categories. The IIRs obtained by GII are less than 0.5 in most cases. Overall, GII outperforms the other approaches in all types of categories.

To determine if the IIR of the existing and novel categories in test samples helps explain the overall NCD and OSR performance, we plot the overall ACC scores of the combination of existing, novel, and unknown categories

($\mathrm{ACC}_{\mathrm{E+N+U}}$) against IIR scores of existing and novel categories (E+N) in Figure 7.4. We find a strong correlation between $\mathrm{ACC}_{\mathrm{E+N+U}}$ scores and IIR (E+N) values, where the Pearson correlation coefficient is -0.63. It indicates that the improvements in IIR can help explain the boost in NCD and OSR performance.

7.4 Conclusion

We have presented a generic one-step representation learning approach to tackle the challenging problem of novel category discovery under an open-set scenario. Our proposed approach consists of three components. First, we achieve intra-class spread for labeled samples by minimizing the intra-class distance. Second, we estimate the novel category centroids and propose intra-cluster loss for the unlabeled samples to discover novel categories. Third, we separate different categories by maximizing the intra-category distance such that all the categories inhabit the same representation space. Last, we evaluated our approach on image and graph datasets, and the results indicate that the proposed approach obtained superior results in NCD and OSR compared with other approaches.

Chapter 8

Conclusion

With the tremendous advances in representation learning, recent machine learning models have achieved great success in various areas. However, these models' success often relies heavily on the massive amount of data collection and human annotations. The real world is an open set, which brings new challenges for machine learning models. First, as new categories emerge on a daily basis, it is difficult to collect samples that cover all the possible categories for training. Thus, an ideal machine learning model should be highly sensitive to unknown categories. Second, it is costly to manually analyze and annotate emerging new categories daily. A mature machine learning model should automatically discover the new categories in the collected training samples. In this dissertation, we focused on solving the above challenges. Specifically, we worked on neural-network-based methods for extracting the representations for open set recognition (OSR) and novel category discovery (NCD).

In Chapter 3, we proposed Min Max Feature (MMF) loss extension to enhance the representation space that leverages intra-class spread and inter-class separation for OSR. The proposed loss extension can be incorporated into different loss functions to find more discriminative representations. We experimented with MMF loss extension on image and graph datasets, and the results indicate that MMF can significantly improve the performances of different types of loss functions.

Chapter 4 to Chapter 5 introduced self-supervised learning approaches as a pre-training stage for OSR. Particularly, we proposed Detransformation Autoencoer (DTAE) in Chapter 4. DTAE engages in learning representations invariant to the input data transformations, and experiments on several standard image datasets indicated that the pre-training process significantly improves the model performance in the OSR tasks. In Chapter 5, we extended DTAE to graph datasets. We proposed two transformations for the function call graph (FCG) based malware representations: FCG-shift and FCG-random. These two transformations are designed to facilitate the pretext task in DTAE. Also, we presented a statistical thresholding approach to find the optimal threshold for the unknown class. The experiment results indicate that our proposed pre-training process can improve the performances of different loss functions for the OSR problem. In Chapter 6, we proposed a feature decoupling approach in the self-supervised pre-training stage. The proposed approach learns a representation that can be split into content and transformation features. Then, only the content features are fine-tuned and used for the OSR problems. We also introduced intra-inter ratio (IIR) to evaluate OSR performance. Moreover, our experimental results indicate that the feature decoupling approach outperforms other self-supervised learning approaches in image and graph OSR problems.

We extended NCD to an open-set setting in Chapter 7. After recognizing the unknown categories, as a following step of OSR, we proposed General Inter-Intra (GII) loss to learn a representation space that clusters the unknown samples under an open-set scenario. Our evaluations show that GII obtains superior results in NCD and OSR compared with other approaches.

8.1 Limitations and Future Work

The open-set problem is a class-incremental learning problem with new categories emerging daily. To maintain a sustainable environment, we need to build a robust and adaptable system that can be continuously updated with samples from new categories without suffering from catastrophic forgetting [65]. The proposed approaches in our work assume all the samples from the old categories are stored in memory and available for training with the new categories. However, it is cost-inhibitive to store all the samples. One way to handle this is to store only representative exemplars for rehearsal, such as iCaRL [73] and AANets [63]. Another way is to prevent consolidated model parameters from drifting via knowledge distillation, such as LwF [62] and ResTune [64].

Moreover, in our novel category discovery approach presented in Chapter 7, we assume the number of novel categories is given. Based on that, we use k-means to estimate the novel categories' centroids. A more practical approach is to estimate the number of novel categories in the unlabelled data first. There are many approaches that can be explored. For example, GCD [89] leverages the information of labeled samples to estimate the number of total categories. They perform k-means on the entire dataset and select the "k" that achieves the highest accuracy on labeled samples as the total number of categories.

Bibliography

[1] Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*, pages 1–8. IEEE, 2020.

[2] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4502–4510, 2016.

[3] Abhijit Bendale and Terrance E. Boult. Towards open set deep networks. In *2016 IEEE Conf. on Computer Vision and Pattern Recognition, CVPR, USA*, pages 1563–1572.

[4] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proc. of the IEEE conf. on computer vision and pattern recognition*, pages 1563–1572, 2016.

[5] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.

[6] Liron Bergman and Yedid Hoshen. Classification-based anomaly detection for general data. In *8th Intl. Conf. on Learning Representations, ICLR 2020*, 2020.

[7] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alexander J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. In *Proc. Thirteenth Intl. Conf. on Intelligent Systems for Molecular Biology 2005, Detroit, MI, USA*, pages 47–56.

[8] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[9] Kaidi Cao, Maria Brbic, and Jure Leskovec. Open-world semi-supervised learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[10] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[11] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 5880–5888. IEEE Computer Society, 2017.

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proc. of the 37th Intl. Conf. on Machine Learning, ICML 2020*, pages 1597–1607.

[13] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 15750–15758. Computer Vision Foundation / IEEE, 2021.

[14] Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík, editors, *Proc. of the Fourteenth Intl. Conf. on Artificial Intelligence and Statistics, AISTATS 2011*, volume 15 of *JMLR Proceedings*, pages 215–223. JMLR.org, 2011.

[15] George E Dahl, Jack W Stokes, Li Deng, and Dong Yu. Large-scale malware classification using random projections and neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3422–3426. IEEE, 2013.

[16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3837–3845, 2016.

[17] Akshay Raj Dhamija, Manuel Günther, and Terrance E. Boult. Reducing network agnostophobia. In *Advances in Neural Information Processing Systems 31*, pages 9175–9186, 2018.

[18] Thomas G. Dietterich. Steps toward robust artificial intelligence. *AI Magazine*, 38(3):3–24, 2017.

[19] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2224–2232, 2015.

[20] Zeyu Feng, Chang Xu, and Dacheng Tao. Self-supervised representation learning by rotation feature decoupling. In *IEEE Conf. on Computer Vision and Pattern Recognition, CVPR 2019, USA*, pages 10364–10374, 2019.

[21] Enrico Fini, Enver Sangineto, Stéphane Lathuilière, Zhun Zhong, Moin Nabi, and Elisa Ricci. A unified objective for novel class discovery. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9264–9272. IEEE, 2021.

[22] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. SCAN: learning to classify images without labels. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part X*, volume 12355 of *Lecture Notes in Computer Science*, pages 268–285. Springer, 2020.

[23] Hugo Gascon, Fabian Yamaguchi, Daniel Arp, and Konrad Rieck. Structural detection of android malware using embedded call graphs. In *AISec'13, Proc. of the 2013 ACM Workshop on Artificial Intelligence and Security*, pages 45–54, 2013.

[24] Zongyuan Ge, Sergey Demyanov, and Rahil Garnavi. Generative openmax for multi-class open set classification. In *British Machine Vision Conf.*, 2017.

[25] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. Recent advances in open set recognition: A survey. *arXiv preprint arXiv:1811.08581*, 2018.

[26] Chuanxing Geng, Sheng-Jun Huang, and Songcan Chen. Recent advances in open set recognition: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(10):3614–3631, 2021.

[27] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *6th Intl. Conf. on Learning Representations, ICLR 2018.*

[28] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

[29] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9781–9791, 2018.

[30] GRAND-Lab. Grand-lab/graph_datasets: A repository of benchmark graph datasets for graph classification (31 graph datasets in total).

[31] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864. ACM, 2016.

[32] Divam Gupta, Ramachandran Ramjee, Nipun Kwatra, and Muthian Sivathanu. Unsupervised clustering using pseudo-semi-supervised learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[33] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[34] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034, 2017.

[35] Kai Han, Sylvestre-Alvise Rebuffi, Sébastien Ehrhardt, Andrea Vedaldi, and Andrew Zisserman. Automatically discovering and learning new visual categories with ranking statistics. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[36] Kai Han, Andrea Vedaldi, and Andrew Zisserman. Learning to discover novel visual categories via deep transfer clustering. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 8400–8408. IEEE, 2019.

[37] Mehadi Hassen and Philip K. Chan. Learning a neural-network-based representation for open set recognition. In *Proc. of the 2020 SIAM Intl. Conf. on Data Mining, SDM, USA*, pages 154–162. SIAM.

[38] Mehadi Hassen and Philip K Chan. Scalable function call graph-based malware classification. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, pages 239–248. ACM, 2017.

[39] Mehadi Hassen and Philip K. Chan. Scalable function call graph-based malware classification. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy,*, pages 239–248, 2017.

[40] Mehadi Hassen and Philip K Chan. Learning to identify known and unknown classes: A case study in open world malware classification. In *The Thirty-First International Flairs Conference*, 2018.

[41] Mehadi Hassen and Philip K. Chan. Learning a neural-network-based representation for open set recognition. *Proc. SIAM Intl. Conf. Data Mining*, pages 154–162, 2020.

[42] Xinwei He, Yang Zhou, Zhichao Zhou, Song Bai, and Xiang Bai. Triplet-center loss for multi-view 3d object retrieval. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1945–1954. Computer Vision Foundation / IEEE Computer Society, 2018.

[43] Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. Deep anomaly detection with outlier exposure. In *7th Intl. Conf. on Learning Representations*, 2019.

[44] Yen-Chang Hsu, Zhaoyang Lv, Joel Schlosser, Phillip Odom, and Zsolt Kira. Multi-class classification without multi-class labels. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[45] Xin Hu, Kang G. Shin, Sandeep Bhatkar, and Kent Griffin. Mutantx-s: Scalable malware clustering based on static features. In *2013 USENIX Annual Technical Conf.*, pages 187–198. USENIX Assoc.

[46] Jingyun Jia and Philip K. Chan. Representation learning with function call graph transformations for malware open set recognition. *Proc. Intl. Joint Conference on Neural Networks, IJCNN 2022 (to appear).*

[47] Jingyun Jia and Philip K. Chan. MMF: A loss extension for feature learning in open set recognition. In *Intl. Conf. on Artificial Neural Networks, Proc. Part II*, pages 319–331, 2021.

[48] Jingyun Jia and Philip K. Chan. Self-supervised detransformation autoencoder for representation learning in open set recognition. *CoRR*, abs/2105.13557, 2021.

[49] Jingyun Jia and Philip K. Chan. Feature decoupling in self-supervised representation learning for open set recognition. *CoRR*, abs/2209.14385, 2022.

[50] Xuhui Jia, Kai Han, Yukun Zhu, and Bradley Green. Joint representation learning and novel category discovery on single- and multi-modal data. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 590–599. IEEE, 2021.

[51] Ming Jin, Yizhen Zheng, Yuan-Fang Li, Chen Gong, Chuan Zhou, and Shirui Pan. Multi-scale contrastive siamese networks for self-supervised graph representation learning. In *Proc. of the Thirtieth Intl. Joint Conf. on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada*, pages 1477–1483.

[52] Inhyuk Jo, Jungtaek Kim, Hyohyeong Kang, Yong-Deok Kim, and Seungjin Choi. Open set recognition by regularising classifier with fake data generated by generative adversarial networks. In *Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2686–2690. IEEE, 2018.

[53] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay S. Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *J. Comput. Aided Mol. Des.*, 30(8):595–608, 2016.

[54] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[55] Nils M. Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. In *Proc. of the 29th Intl. Conf. on Machine Learning, Edinburgh, Scotland, UK*. icml.cc / Omnipress, 2012.

[56] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[57] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[58] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST database, 1999.

[59] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *6th Intl. Conf. on Learning Representations*, 2018.

[60] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

[61] Xiaoli Li and Bing Liu. Learning from positive and unlabeled examples with different data distributions. In *European Conf. on Machine Learning*, pages 218–229, 2005.

[62] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947, 2018.

[63] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 2544–2553. Computer Vision Foundation / IEEE, 2021.

[64] Yu Liu and Tinne Tuytelaars. Residual tuning: Toward novel category discovery without labels. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2022.

[65] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.

[66] Lawrence Neal, Matthew Olson, Xiaoli Fern, Weng-Keen Wong, and Fuxin Li. Open set learning with counterfactual images. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 613–628, 2018.

[67] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conf, Amsterdam, The Netherlands, Proc., Part VI*, volume 9910 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2016.

[68] Enrique G. Ortiz and Brian Christopher Becker. Face recognition for web-scale datasets. *Comput. Vis. Image Underst.*, 118:153–170, 2014.

[69] Poojan Oza and Vishal M. Patel. C2AE: class conditioned auto-encoder for open-set recognition. In *Conf. on Computer Vision and Pattern Recognition, 2019*, pages 2307–2316.

[70] Xi Peng, Xiang Yu, Kihyuk Sohn, Dimitris N. Metaxas, and Manmohan Chandraker. Reconstruction-based disentanglement for pose-invariant face recognition. In *IEEE International Conference on Computer Vision, ICCV 2017*, pages 1632–1641.

[71] Pramuditha Perera et al. Generative-discriminative feature representations for open-set recognition. In *2020 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 11811–11820.

[72] Stanislav Pidhorskyi, Ranya Almohsen, and Gianfranco Doretto. Generative probabilistic novelty detection with adversarial autoencoders. In *Annual Conf. on Neural Information Processing Systems 2018, NeurIPS 2018, Montréal, Canada*, pages 6823–6834.

[73] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5533–5542. IEEE Computer Society, 2017.

[74] Royi Ronen, Marian Radu, Corina Feuerstein, Elad Yom-Tov, and Mansour Ahmadi. Microsoft malware classification challenge. *CoRR*, abs/1802.10135, 2018.

[75] Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Björn Ommer, and Joseph Paul Cohen. Revisiting training strategies and generalization performance in deep metric learning. In *Proc. of the 37th Intl. Conf. on Machine Learning, ICML 2020, Virtual Event*, volume 119, pages 8242–8252. PMLR.

[76] Andras Rozsa, Manuel Günther, and Terrance E. Boult. Adversarial robustness: Softmax versus openmax. In *British Machine Vision Conf. 2017*, 2017.

[77] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 153–168, 2018.

[78] Walter J. Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E. Boult. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, 2013.

[79] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *Information Processing in Medical Imaging - 25th Intl. Conf.*, pages 146–157, 2017.

[80] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Conf. on Computer Vision and Pattern Recognition*, pages 815–823. IEEE, 2015.

[81] Alexander Schultheiss, Christoph Käding, Alexander Freytag, and Joachim Denzler. Finding the unknown: Novelty detection with extreme value signatures of deep neural activations. In *Pattern Recognition - 39th German Conference*, pages 226–238, 2017.

[82] Matthew G. Schultz, Eleazar Eskin, Erez Zadok, and Salvatore J. Stolfo. Data mining methods for detection of new malicious executables. In *2001 IEEE Symposium on Security and Privacy*, pages 38–49. IEEE Computer Society.

[83] Kristof T. Schütt, Stefan Arbabzadah, Farhad Chmiela, Klaus R. Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature Communications*, 8, 2017.

[84] Lei Shu, Hu Xu, and Bing Liu. Unseen class discovery in open-world classification. *arXiv preprint arXiv:1801.05609*, 2018.

[85] Yu Shu, Yemin Shi, Yaowei Wang, Yixiong Zou, Qingsheng Yuan, and Yonghong Tian. ODN: Opening the deep network for open-set action recognition. In *2018 IEEE Intl. Conf. on Multimedia and Expo (ICME)*, pages 1–6.

[86] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[87] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[88] Li Tang, Yue Wang, Qianhui Luo, Xiaqing Ding, and Rong Xiong. Adversarial feature disentanglement for place recognition across changing appearance. In *2020 IEEE Intl. Conf. on Robotics and Automation, ICRA*, pages 1301–1307.

[89] Sagar Vaze, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Generalized category discovery. *CoRR*, abs/2201.02609, 2022.

[90] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 1096–1103. ACM, 2008.

[91] Zhuoyi Wang, Zelun Kong, Hemeng Tao, Swarup Chandra, and Latifur Khan. Co-representation learning for classification and novel class detection via deep networks. *arXiv preprint arXiv:1811.05141*, 2018.

[92] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VII*, volume 9911 of *Lecture Notes in Computer Science*, pages 499–515. Springer, 2016.

[93] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[94] Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. Infogcl: Information-aware graph contrastive learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 30414–30425. Curran Associates, Inc., 2021.

[95] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 5449–5458, 2018.

[96] Jay Yagnik, Dennis Strelow, David A. Ross, and Ruei-Sung Lin. The power of comparative reasoning. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc Van Gool, editors, *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 2431–2438. IEEE Computer Society, 2011.

[97] Pinar Yanardag and S. V. N. Vishwanathan. Deep graph kernels. In *Proc. of the 21th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, Australia, 2015*, pages 1365–1374.

[98] Muli Yang, Yuehua Zhu, Jiaping Yu, Aming Wu, and Cheng Deng. Divide and conquer: Compositional experts for generalized novel class discovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14268–14277, June 2022.

[99] Ryota Yoshihashi, Wen Shao, Rei Kawakami, Shaodi You, Makoto Iida, and Takeshi Naemura. Classification-reconstruction learning for open-set recognition. In *Conf. on Computer Vision and Pattern Recognition*, pages 4016–4025, 2019.

[100] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Annual Conf. on Neural Information Processing Systems, NeurIPS 2020, virtual*.

[101] Yang Yu, Wei-Yang Qu, Nan Li, and Zimin Guo. Open category classification by adversarial sample generation. In *Proc. of the Twenty-Sixth Intl. Joint Conf. on Artificial Intelligence*, pages 3357–3363, 2017.

[102] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In Marina Meila and Tong Zhang, editors, *Proc. of the 38th Intl. Conf. on Machine Learning, ICML 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 12310–12320. PMLR.

[103] Hao Zhang, Mengmeng Wang, Yong Liu, and Yi Yuan. FDN: feature decoupling network for head pose estimation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 12789–12796. AAAI Press, 2020.

[104] Bingchen Zhao and Kai Han. Novel visual category discovery with dual ranking statistics and mutual knowledge distillation. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 22982–22994, 2021.

[105] Zhun Zhong, Enrico Fini, Subhankar Roy, Zhiming Luo, Elisa Ricci, and Nicu Sebe. Neighborhood contrastive learning for novel class discovery. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 10867–10875. Computer Vision Foundation / IEEE, 2021.

[106] Zhun Zhong, Linchao Zhu, Zhiming Luo, Shaozi Li, Yi Yang, and Nicu Sebe. Openmix: Reviving known knowledge for discovering novel visual categories in an open world. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 9462–9470. Computer Vision Foundation / IEEE, 2021.

[107] Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Learning placeholders for open-set recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 4401–4410.

[108] Yajin Zhou and Xuxian Jiang. Android malware genome project, 2015.