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

Jingyun Jia*†, Philip K. Chan*
* Florida Institute of Technology
† Baidu Research

***Abstract*—Assuming unknown classes could be present during classification, the open set recognition (OSR) task aims to classify an instance into a known class or reject it as unknown. In this paper, we use a two-stage training strategy for OSR problems. In the first stage, we introduce a self-supervised feature decoupling method that finds the content features of the input samples from the known classes. Specifically, our feature decoupling approach learns a representation that can be split into content features and transformation features. In the second stage, we fine-tune the content features with the class labels. The fine-tuned content features are then used for the OSR problems. To measure representation quality, we introduce intra-inter ratio (IIR). Our experimental results indicate that our proposed self-supervised approach outperforms others in image and malware OSR problems. Also, our analyses indicate that IIR is correlated with and can explain OSR performance.**

***Index Terms*—Open set recognition; self-supervised learning; unsupervised learning; representation learning**

I. 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) [1]. In this paper, 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 that involve transformations [2]–[4]. 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. We introduce a feature decoupling approach during the first (pre-training) stage to extract the content features irrelevant to transformation information. Our approach decouples the representations through content reconstruction and transformation prediction tasks. We achieve this goal by reconstructing the different views to their original form. Furthermore, the transformation features should contain 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. In the second stage, we fine-tune the content features with the provided class labels and discard the transformation features. Finally, the fine-tuned content features are used for the OSR tasks.

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, 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.

II. Related Work

An **Open Set Recognition (OSR)** task has two objectives: classify the known classes and recognize the unknown class which is absent from training. We divide recent research into three categories. Approaches in the first category borrow other data samples used as the unknown class. Dhamija et al. [5] utilize the differences in feature magnitudes between known classes. They borrow unknown samples as part of the objective function. The approaches in the second category generate additional data as the unknown class. Ge et al. [6] introduce a conditional GAN to generate some unknown samples followed by an OpenMax classifier. The third category of OSR approaches does not borrow nor generate additional data as the unknown class in training. Bendale and Boult [7] propose OpenMax for OSR. OpenMax adapts Meta-Recognition concepts to the activation patterns in the representation layer of the network and then estimates the probability of an input being from an unknown class. Hassen and Chan [8] 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. Jia and Chan [9] propose MMF as a loss extension to further separate the known and unknown representations for OSR.

Self-supervised learning learns representations via pretext tasks that are different from the primary tasks. Autoencoding is a simple example. It learns the representation by reconstructing the original input samples. Besides autoencoder, most research works introduce these pretext tasks by transforming the original inputs into different views. Recent works have applied a self-supervised learning approach to

various domains. For example, in image representation learning, Gidaris et al. [2] propose RotNet, where the pretext task predicts the rotation degree. Zbontar et al. [3] 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. Moreover, Jia and Chan [4] propose DTAE, which reconstructs the different views back to their original forms. Thus, the representations of different views of the same input should be similar.

Besides image representation learning, more recent works have extended self-supervision approaches to graph representation learning. For example, You et al. [10] 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. Jia and Chan [11] extend the DTAE framework to function call graphs (FCGs) by introducing two FCG transformations: FCG-shift and FCG-random. Moreover, Jin et al. [12] 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.

Some research proposes **feature disentanglement and decoupling** to improve representation learning. Feng et al. [13] 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. [14] propose a feature disentanglement approach for face recognition, where they introduce a feature reconstruction metric learning to disentangle identity and pose features. To estimate head pose from RGB images, Zhang et al. [15] introduce a feature decoupling module to learn the discriminative features for each pose angle explicitly. Our proposed approach does not produce degenerate content features (vs. rotational irrelevance [13]) since they are used to reconstruct the original instances. The reconstruction is also more constrained than rotation irrelevance and image instance classification [13] for potentially more meaningful representations. Furthermore, besides images, we use and evaluate our approach on graphs.

III. 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 use 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.

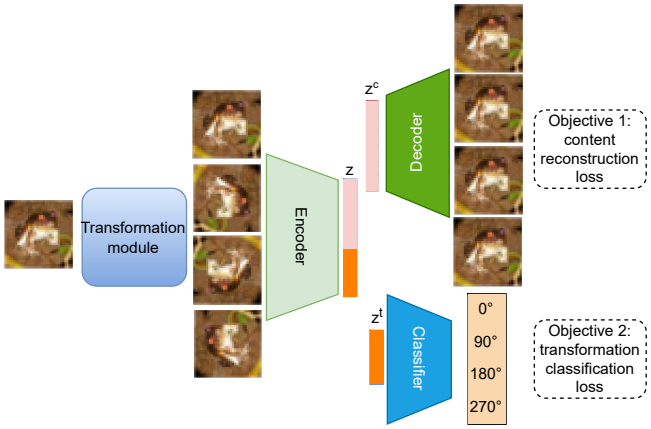


Figure 1: Illustration of proposed feature decoupling method. The transformation module transforms the original input samples into several correlation views. The encoder outputs decoupled content and transformation features. The content part is learned by reconstructing the transformed input samples back to their original forms, and the transformation part is learned by transformation classification.

A. 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 1, given an original input sample x, we use a transformation module T to augment the input x with several different correlated views. The transformation module contains M different transformations as $T = \{t_1, t_2, ...t_m\}$. In our example in Figure 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 the content features of the transformed data example. In contrast, 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.

1) Learning content features: The content features should be invariant for all its transformed views for the same input sample. Here, we apply a content reconstruction decoder on reconstructed transformed samples. As shown in the "Objective 1" in Figure 1, the input of the decoder is the content part of the representation, z_j^c. Moreover, instead of reconstructing the content features back to their transformed views, the decoder here "reconstructs" them to their original form before the transformation module.

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

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

Each data point has M transformations, and there are M times data points as the original input sample after the transformation module.

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 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:

$$\mathcal{L}_{\text{transformation}} = -\log p_{i=j}(g^t(z_j^t)), \tag{2}$$

where j is the ground truth transformation label of input sample x_j. In our example in Figure 1, the objective of the classifier is to classify four rotation types introduced from the transformation module.

B. Fine-tuning stage

In the first stage, the self-supervised feature decoupling approach 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 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 classification loss, representation loss functions do not require a classifier. They constrain the representation layers directly, such as triplet loss [16]. Figure 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. During the inference time, we only use the content features z^c to represent the input sample for the OSR tasks.

C. Open set recognition

After the second stage, we obtain the encoder and the centroids of 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 centroids (μ_k) of all the known classes:

$$outlier\_score(x) = \min_{1\le k\le K} \|\mu_k - z^c\|_2^2 \tag{3}$$

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 which known class the test sample belongs. 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:

$$P(y=k|x) = \frac{e^{-\|\mu_k - z^c\|_2^2}}{\sum_{k=1}^{K} e^{-\|\mu_k - z^c\|_2^2}} \tag{4}$$

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

$$\hat{y} = \begin{cases} unknown, & \text{if } outlier\_score(x) > t \\ \underset{1\le k\le K}{\operatorname{argmax}} P(y=k|x), & \text{otherwise} \end{cases} \tag{5}$$

$$\mu_k = \frac{1}{N_k}\sum_{i=1}^{N_k} z_i^c \tag{6}$$

IV. Experiments

We evaluate the proposed feature decoupling approach with two types of fine-tuning functions as mentioned in Section III-B: classification loss (cross-entropy loss) and

(a) Fine-tuning with classification loss

(b) Fine-tuning with representation loss

Figure 2: Illustration of two types of fine-tuning objectives. Only the original input sample is used in the fine-tuning stage. The encoder and representation layer are inherited from the pre-training stage. 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.

representation loss (triplet loss). Moreover, to show that our proposed approach works on different datasets. We test the approach on images and malware datasets.

Fashion-MNIST [17] 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 [18] 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) [19] 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 [20] 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 [21]. 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 [22] to extract the function instructions and then generated the FCGs as in [20]. Also, to simulate an open-set scenario, we randomly pick six digits as the known classes while considering the rest as the unknown class.

A. 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 [2], Barlow Twins [3], DTAE [4] [23]. We construct a fine-tuning network to refine the learned content features in the second stage. 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 [1].

As illustrated in Figure 1, 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 [23] 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. We experimented with different dimensional combinations of the content features and the transformation features. The setting that uses six dimensions for the content features and three dimensions for the non-content transformation features performs better across datasets. The six-dimensional content features are connected to a decoder, which is simply the reverse of the encoder. The three-dimensional transformation features are 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's keep probability 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 [2], 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 [3]. Here, we apply the same constraints to the cross-correlation matrices between the transformed views and their corresponding original samples.

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 2a or a representation loss function as shown in Figure 2b. 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, and then the representation is directly fed to a softmax layer.

B. Evaluation criteria

To simulate an open-set scenario, we randomly pick six classes as known classes and use them in the training process. The ensemble of the remaining classes is considered the unknown class, which does not participate in the training process and only exists in the test set. Specifically, 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 decoupling (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. We measure both ROC AUC scores under 100% and 10% FPRs. The ROC AUC scores under 100% FPR are 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.

C. Evaluation results

Table I 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 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). Significantly, 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 II. 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). Like the AUC ROC results, OpenMax outperforms cross-entropy 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, our proposed self-supervised feature decoupling approach in most cases 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 I and Table II. 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 I and Table II indicate that

Table I: 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. The values in bold are the highest in each group.

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

Table II: 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). The values are the highest in each group.

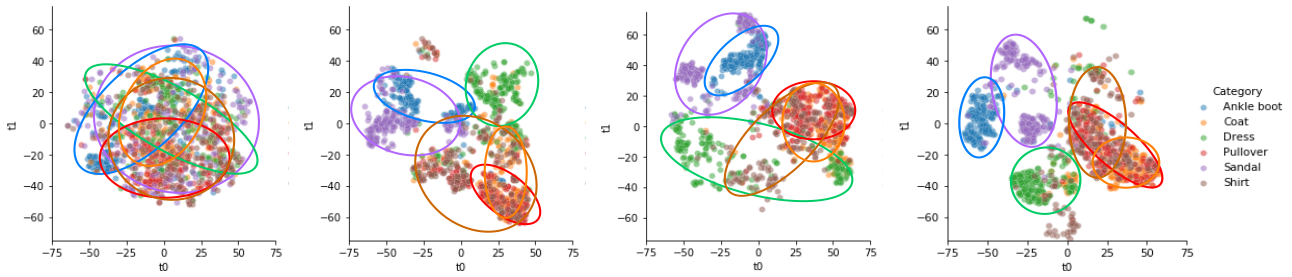
Image Dataset		Fashion-MNIST			CIFAR-10		
		Known	Unknown	Overall	Known	Unknown	Overall
OpenMax		0.747±0.049	0.521±0.178	0.714±0.051	0.645±0.022	0.540±0.065	0.630±0.017
ce	No Pre-training	0.685±0.102	0.559±0.076	0.667±0.086	0.567±0.048	0.369±0.169	0.538±0.045
	RotNet	0.711±0.067	0.569±0.097	0.691±0.058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

the transformation classifier usually contributes to improved performance in our proposed approach.

D. Analysis of the self-supervised models

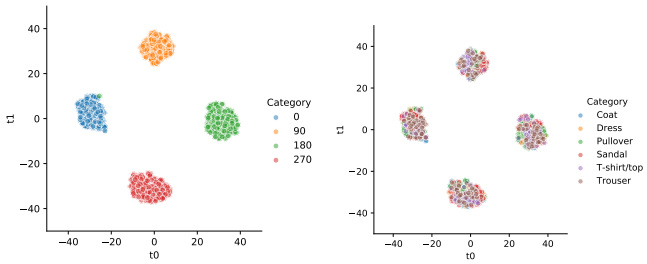
Our experiment results indicate that the proposed feature decoupling approach benefits different loss functions on the OSR tasks. We plot the t-SNE plots at different stages to further analyze the model performances. Figure 3 shows the t-SNE plots of the known classes in the (unseen) test set of Fashion-MNIST after pre-training. 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.

To show that the learned transformation features do not contain class information, we color the transformation features of Fashion-MNIST samples with respect to transformation labels and fashion labels in Figure 4. If the rep-



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

Figure 3: The t-SNE plots of the representations of Fashion-MNIST test samples on pre-trained models.



(a) With transformation labels (b) With class labels

Figure 4: The t-SNE plots of the learned transformation features for the Fashion-MNIST test samples on the pre-trained models

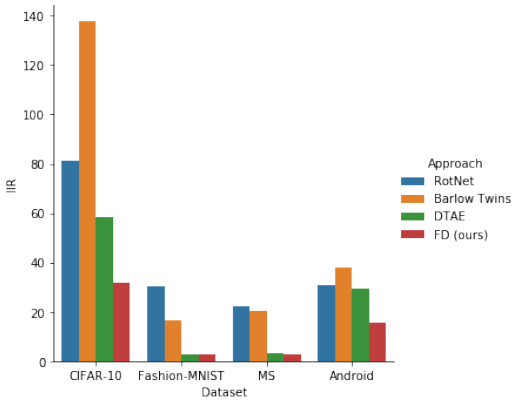


Figure 5: IIR after the pre-training stage.

resentations have class information, representations in the same class will be clustered together; otherwise, they will be dispersed. We observe that transformation features contain mostly transformation information as in Figure 4a and little class information as shown in Figure 4b.

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:

$$intra_k = \frac{1}{N_k} \sum_{i=1}^{N_k} d(\mu_k, z_i), \tag{7}$$

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

$$inter_k = \min_{i, i \neq k} d(\mu_k, \mu_i) \tag{8}$$

Moreover, the intra-inter ratio of class k is defined as $IIR_k = intra_k / inter_k$. The denominator $inter_k$ serves as a normalization factor so that we can compare the representation quality across different approaches. Here, we use the average IIR over the K known classes to measure the overall representation quality:

$$IIR = \frac{1}{K} \sum_{k=1}^{K} \frac{intra_k}{inter_k} \tag{9}$$

IIR is similar to the feature space density proposed by Roth et al. [24]. One difference is that IIR calculates the average ratio of all classes instead of the average intra-distance and inter-distance ratio. 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 5 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 the initial IIR before the training process can be infinite when the representations of instances are randomly distributed. Moreover, 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.

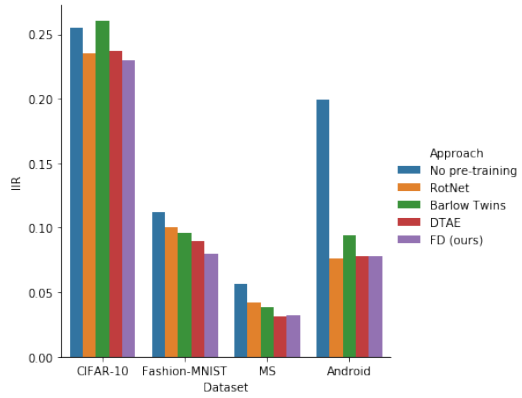


Figure 6: IIR after the fine-tuning stage.

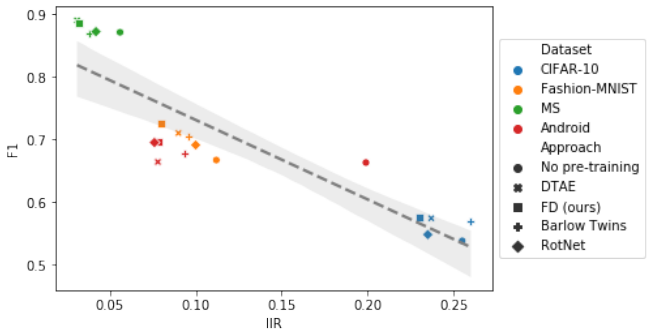


Figure 7: F1 against IIR after the fine-tuning stage.

E. Analysis of the fine-tuned models

We plot the IIR of the models fine-tuned by cross-entropy loss in Figure 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 II against IIR in Figure 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.

V. Conclusion

We use a two-stage learning approach for OSR problems. In the first stage, we propose a self-supervised feature decoupling method to split the learned representation into the content and transformation parts. In the second stage, we fine-tune the content features from the first stage with class labels. 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 OSR scenarios with image and malware datasets. Our analyses indicate that IIR is correlated with and can explain OSR performance.

References

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

[2] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *6th Intl. Conf. on Learning Representations, ICLR 2018*.

[3] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," in *Proc. of the 38th Intl. Conf. on Machine Learning, ICML 2021, Virtual Event*, vol. 139. PMLR, pp. 12 310–12 320.

[4] J. Jia and P. K. Chan, "Self-supervised detransformation autoencoder for representation learning in open set recognition," in *ICANN - 31st Intl. Conf. on Artificial Neural, UK*, vol. 13532. Springer, 2022, pp. 471–483.

[5] A. R. Dhamija, M. Günther, and T. E. Boult, "Reducing network agnostophobia," in *Advances in Neural Information Processing Systems 31*, 2018, pp. 9175–9186.

[6] Z. Ge, S. Demyanov, and R. Garnavi, "Generative openmax for multi-class open set classification," in *British Machine Vision Conf.*, 2017.

[7] A. Bendale and T. E. Boult, "Towards open set deep networks," in *2016 IEEE Conf. on Computer Vision and Pattern Recognition, CVPR, USA*, pp. 1563–1572.

[8] M. Hassen and P. K. Chan, "Learning a neural-network-based representation for open set recognition," *Proc. SIAM Intl. Conf. Data Mining*, pp. 154–162, 2020.

[9] J. Jia and P. K. Chan, "MMF: A loss extension for feature learning in open set recognition," in *Intl. Conf. on Artificial Neural Networks, Proc. Part II*, 2021, pp. 319–331.

[10] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Annual Conf. on Neural Information Processing Systems, NeurIPS 2020, virtual*.

[11] J. Jia and P. 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)*.

[12] M. Jin, Y. Zheng, Y. Li, C. Gong, C. Zhou, and S. 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*, pp. 1477–1483.

[13] Z. Feng, C. Xu, and D. Tao, "Self-supervised representation learning by rotation feature decoupling," in *IEEE Conf. on Computer Vision and Pattern Recognition, CVPR 2019, USA*, 2019, pp. 10 364–10 374.

[14] X. Peng, X. Yu, K. Sohn, D. N. Metaxas, and M. Chandraker, "Reconstruction-based disentanglement for pose-invariant face recognition," in *IEEE Intl. Conf. on Computer Vision, ICCV 2017*, pp. 1632–1641.

[15] H. Zhang, M. Wang, Y. Liu, and Y. Yuan, "FDN: feature decoupling network for head pose estimation," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press, 2020, pp. 12 789–12 796.

[16] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Conf. on Computer Vision and Pattern Recognition*. IEEE, 2015, pp. 815–823.

[17] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[18] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[19] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft malware classification challenge," *CoRR*, vol. abs/1802.10135, 2018.

[20] M. Hassen and P. K. Chan, "Scalable function call graph-based malware classification," in *Proc, of the Seventh ACM on Conf. on Data and Application Security and Privacy,*, 2017, pp. 239–248.

[21] Y. Zhou and X. Jiang, "Android malware genome project," 2015. [Online]. Available: http://www.malgenomeproject.org/

[22] H. Gascon, F. Yamaguchi, D. Arp, and K. Rieck, "Structural detection of android malware using embedded call graphs," in *AISec'13, Proc. of the 2013 ACM Workshop on Artificial Intelligence and Security*, pp. 45–54.

[23] J. Jia and P. K. Chan, "Representation learning with function call graph transformations for malware open set recognition," in *Intl. Joint Conf. on Neural Networks (IJCNN)*, 2022, pp. 1–8.

[24] K. Roth, T. Milbich, S. Sinha, P. Gupta, B. Ommer, and J. P. 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*, vol. 119. PMLR, pp. 8242–8252.