York University

EECS - 6412 Data Mining

Project Report

Multi Radius Deep One-Class Classification

Group Members:

Shreejal Trivedi

Sharuka Thirimanne

December 27, 2022

Abstract

Our work is the complete extension of the work by **[5] - Deep SVDD: Deep One Class Classification**. We observed some trends in different scenarios where this paper can be utilized, and all our observations are derived from that. Apart from our observation and changes, all the courtesy is to the authors [5]. We try to extend the work done by the authors to multi-radius anomaly deep one class classification where we can cluster the embedding space to generate the pseudo-labels of the clusters and then use those class information for further learning the radii for each cluster instead of unique radius for all the classes. This method helps use to get more compact radii for each class and during the inference, we make the decision on the data to be an anomaly, if it is an outlier for all the classes. We have shown the proof of concept on the two well-known datasets of CIFAR10 and MNIST. Due to GPU and time constraints, we have mentioned anomaly detection results only for several classes on both the datasets. We developed our code on the original PyTorch implementation of the authors: `https://github.com/lukasruff/Deep-SVDD-PyTorch`

0 Disclaimer: Extra mentions and analysis of the algorithm

All the theorems and the theoretical analysis of the algorithms are taken from the original paper by **Ruff et.al**[5]. We have just extended the DOC objective to multi-radius. If you want to learn about the theoretical analysis of the theorems you can read the original **research paper**[**5**]. We would like to thank the original authors of this paper for a thorough development of the algorithm and ablation studies.

1 Introduction

Anomaly detection is a technique that has been widely used in fields where finding anomalous data is complex, such as intrusion detection, fraud detection, and medical diagnosis. For instance, collecting data that represents both 'normal' and 'anomalous' for intrusion detection by employing data packet capturing mechanisms to collect data sequences is difficult. Since patterns that represent intrusions or anomalous are scarce in daily network traffic. Anomaly detection uses unusual discerning samples in data, and it is considered an unsupervised learning method.

With the increasing popularity of computer vision tasks, in recent works, many advancements are done in anomaly detection at the image level, which considers some generative modeling and auto-encoder-based approaches to solve the problem. However, the work which took the original anomaly detection objecting function to learn anomalies based on the image representations is scarce. One-Class SVM and Kernel Density Estimation (KDE) are popular anomaly detection techniques. However, those suffering from several inherent problems, such as often failures in high-dimensional datasets due to the curse of dimensionality and poor computational scalability. To overcome that, substantial feature engineering is required. By employing deep learning, it is possible to learn relevant features automatically. Training a deep neural network (DNN) while minimizing the volume of a hypersphere that encloses the

network representations of the data will force the network to extract the common factors of variations.

In previous studies, Deep Convolutional Neural Networks (DCNNs) were employed to learn the deep representations by incorporating the radius of the hypersphere. Even though it provides a compact cluster or the region which captures the typical variations in the representations, the authors have used a single radius for all the classes. In an actual scenario, this might make a difference because different classes have different compactness and variations, which might not be captured by a single volume or the radius of the hypersphere. We would like to extend previous work in this study by considering the class information and learning the class-conditional hypersphere. This might help us in capturing the class-conditional anomalies perfectly. This can be done by taking into account the clusters of the particular class and minimizing the total objective function of the classification loss and the anomaly detection loss per class instead of taking the single dataset as a whole.

2 Related Work

Anomaly detection is an extensively studied area in time series analysis. But as we focus on image-level anomaly detection. Various traditional algorithms such as Kernel based anomaly detection. The most widely used algorithm used in anomaly detection is by Scholkopf et.al [8]. In this paper, the author tries to find the best separating hyperplane in the space where the inliers are in a compact space and outliers are far away from the clusters. Another method by Tax et.al[9], where instead of finding the hyperplane, the author tries to find the hypersphere by minimizing the radius $\mathcal{R}$ with the center c. SVDD[9] can be considered as a non-linear version of the OC-SVM[8] and their dual representations can be solved with similar methods. But these kernel-based methods are not efficient in scalability, and they scale quadratically on the amount of data. Also, to get accurate results, we might need to increase the number of support vectors involved in the decision-making, increasing the memory requirements.

However, with the advent of deep learning-based techniques, such as Convolutional Neural Networks, have been used extensively in computer vision tasks to learn powerful and especially important features of images. Therefore, researchers are now focusing on using the CNNs and their variations of Fully Convolutional Neural Neworks for deep anomaly detection. Some of them are Generative Adversarial Networks[1] and Autoencoders [3]. Autoencoder is a variation of Fully Convolutional Neural Networks, whose objective function is to learn the compressed representation of an image. This is accomplished using the reconstruction loss $||x - x^2||$. The problem of using the autoencoders in anomaly detection directly is finding the optimal compression code of an image(or an intermediate representation) and learning the compactness of the compressed representations. But the authors of deep SVDD incorporate to use of the autoencoders in a different way. They first learn the intermediate representations using the AEs and incorporate those in the DOC objective function. We extend this work to a multi-radius DOC objective function. The theorems can be found in the section on the **Approach**.

The paper by Schelegl et.al [7] tries to incorporate GANs in AD. They mention that the outliers might not have a good representation, unlike the normal examples. They try to

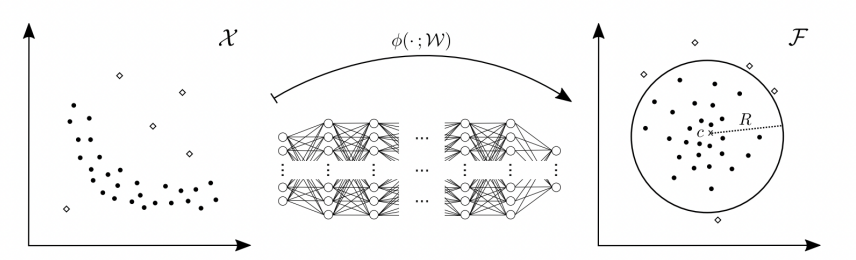

Figure 1: Taken from **[5]**

learn the GAN using the examples present in the training data, and the generator tries to generate as near as good examples as in the training data. But for an outlier, it will be hard for the generator to induce the examples that will differ from the inliers. They incorporate the reconstruction loss during the testing to get the score that defines if the data point is normal or an anomaly. But the same problem persists in the approach of training GANs i.e., how to incorporate the compactness factor during the training as the more compact the clusters are, the better the SVDD will learn the radius with respect to the cluster centroids.

3 Approach

In this paper, the authors try to minimize the radius of the hypersphere by making the space compact as most common features or the images should be close to the center of the sphere. They try to leverage the One-Class Support Vector Machine's objective function in deep neural networks. All the results that were shown in the paper are based only on the one class i.e. they consider only one center and radius of the sphere and try to minimize the compactness of the cluster. But this algorithm does not give good results when we try to mix more than one class and learn just the one radius for all of these classes. Basically, in an unsupervised setting, we don't know the data labels, so there can be data that are organized in more than one cluster. So learning one radius for each cluster can become biased to the cluster having a high number of data points.

As mentioned earlier, we extend the work by Ruff et.al [5] and add the class conditional radius learning on the top of the DOC objective. The whole pipeline is shown fig. 2. For generating the pseudo-labels, we incorporate the algorithm by Sarfraz et.al [6].

As shown in Fig.2(left), we add an extra layer of training pipeline that takes the embeddings learned in stage 1 and generates pseudo labels using the FINCH[6] algorithm. We use these cluster assignments in optimizing our extended DOC objective function.

3.1 Theorems

This section states the theorems that the authors are trying to optimize using the DeepSVDD in [6]. **All the theorems that are stated below are directly taken from the paper.**. Then, we present our objective function for multi-radius learning.

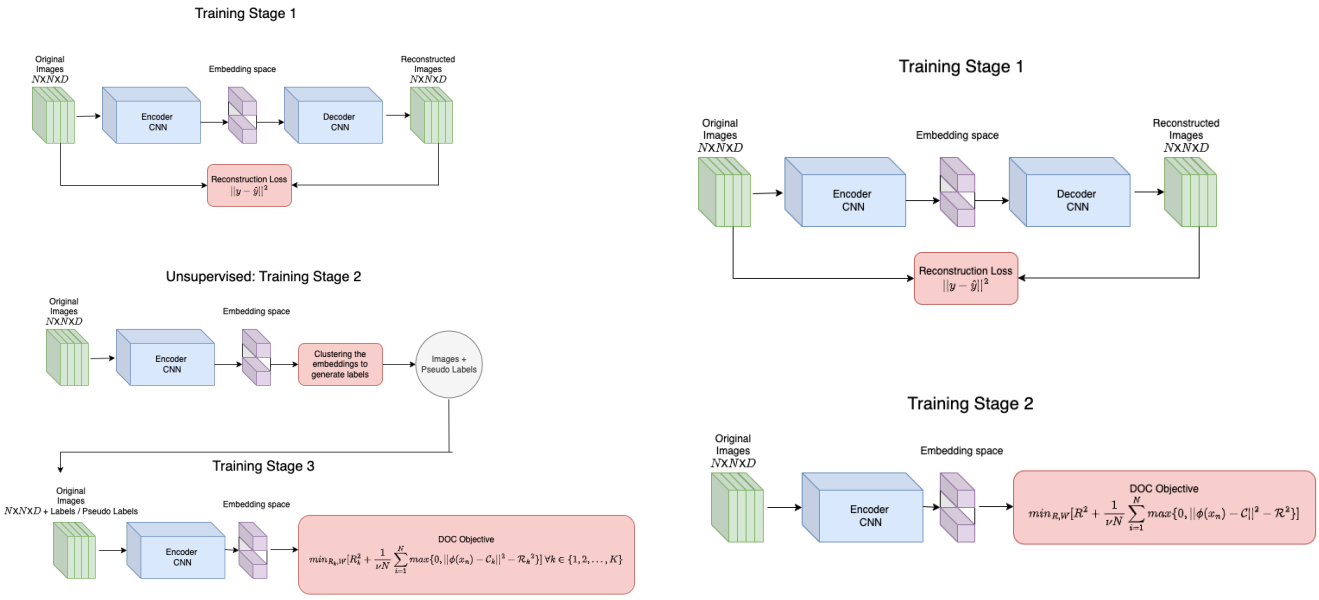

Figure 2: Left. Proposed pipeline. Right. Original pipeline as in Ruff et.al [5]

Theorem 1 (Theorem 1: Deep SVDD Objective Function ([5])) *The objective of the SVDD is to find the smallest hypersphere with center $c \in \mathcal{F}_k$ and radius $R > 0$ that encloses the majority of the data in the feature space $\mathcal{F}_k$*

$$\min_{\mathcal{R},c,\zeta} \mathcal{R}^2 + \frac{1}{\nu n} \sum_i \zeta_i \tag{1}$$

$$||\phi_k(x_i) - c^2||_{\mathcal{F}k} \leq \mathcal{R}^2 + \zeta_i, \forall i$$

Theorem 2 (Theorem 2: Soft Boundary Deep SVDD Objective Loss Function ([5])) *Consider the input space $\mathcal{X} \subseteq \mathcal{R}^D$ and the output space $\mathcal{F} \subseteq \mathcal{R}^P$. We also have network $\phi(.;\mathcal{W}) : \mathcal{X} \to \mathcal{F}$ with $L \in \mathcal{N}$ layers and set of weights $\mathcal{W}$. Soft boundary Deep SVDD objective is to minimize the data enclosing hypersphere using center c and radius $R > 0$ using the training data $\mathcal{D}_n = \{x_1, x_2, ..., x_n\}$*

$$\min_{\mathcal{R},\mathcal{W}} \mathcal{R}^2 + \frac{1}{\nu n} \sum_i^n max\{0, ||\phi(x_i, \mathcal{W}) - c||^2 - \mathcal{R}^2\} \tag{2}$$

Both the theorems are very similar in structure which helps to learn a hypersphere that contains most of the normal examples. In **Theorem 2**, the authors mention that if we minimize the $\mathcal{R}^2$ as we do in SVDD, we are, in fact, minimizing the radius of the hypersphere, which will help us to learn the important variations of the normal examples and will neglect the high and non-transient data statistics. The parameter $\nu \in [0, 1]$ is a hyperparameter that controls the tradeoff between the penalty term and the radius of the hypersphere. If

the value of ν is high, more points outside the radius will be considered during the training. In other words, it tells us about the proportion of anomalous points.

Whereas, in **Theorem 1**, the objective function just tries to minimize the mean distance of the points from the centroid of the cluster, unlike the soft-boundary objective function where the sphere is contracted by minimizing the radius of the sphere. We can consider *Theorem 2* as a soft version of the one-class SVDD.

Theorem 3 (Extended objective function for Deep SVDD - Theorem 2) *Consider the input space $\mathcal{X} \subseteq \mathcal{R}^D$ and the output space $\mathcal{F} \subseteq \mathcal{R}^P$. We also have network $\phi(.;\mathcal{W}) : \mathcal{X} \to \mathcal{F}$ with $L \in \mathcal{N}$ layers and set of weights $\mathcal{W}$. Soft boundary Deep SVDD objective is to minimize the data enclosing hypersphere using center c_k and radius $R_k > 0$ using the training data $\mathcal{D}_n = \{x_1, x_2, ..., x_n\}$ for class $k \in \{1, 2, ..., K\}$*

$$\min_{\mathcal{R}_k, \mathcal{W}} \mathcal{R}_k^2 + \frac{1}{\nu n} \sum_{i}^{n} max\{0, ||\phi(x_i, k_i, \mathcal{W}) - c_k||^2 - \mathcal{R}_k^2\} \tag{3}$$

Here, k is the number of classes we get after clustering the input embeddings. Mostly hierarchical clustering such as FINCH([6]) is used to obtain the clusters. These clusters are then used as pseudo labels to generate the cluster centroids. These centroids are helping the radius $\mathcal{R}_k$ to adjust to their respective compact space. ***In this project, we focus on the soft-boundary objective function of minimizing the radius of the hypersphere***. During the inference, data point $x \in \mathcal{X}$ we classify the point as an anomaly if $\arg\min_{\mathbf{k}} ||\phi(\mathbf{x}; \mathcal{W}) - \mathbf{c_k}||^{\mathbf{2}} - \mathcal{R}_{\mathbf{k}}^{\mathbf{2}}$ $\forall k \in \{1, 2, .., K\}$ turns out to be positive. This shows that if the point does not fall in any cluster, then the point has to be declared an anomaly.

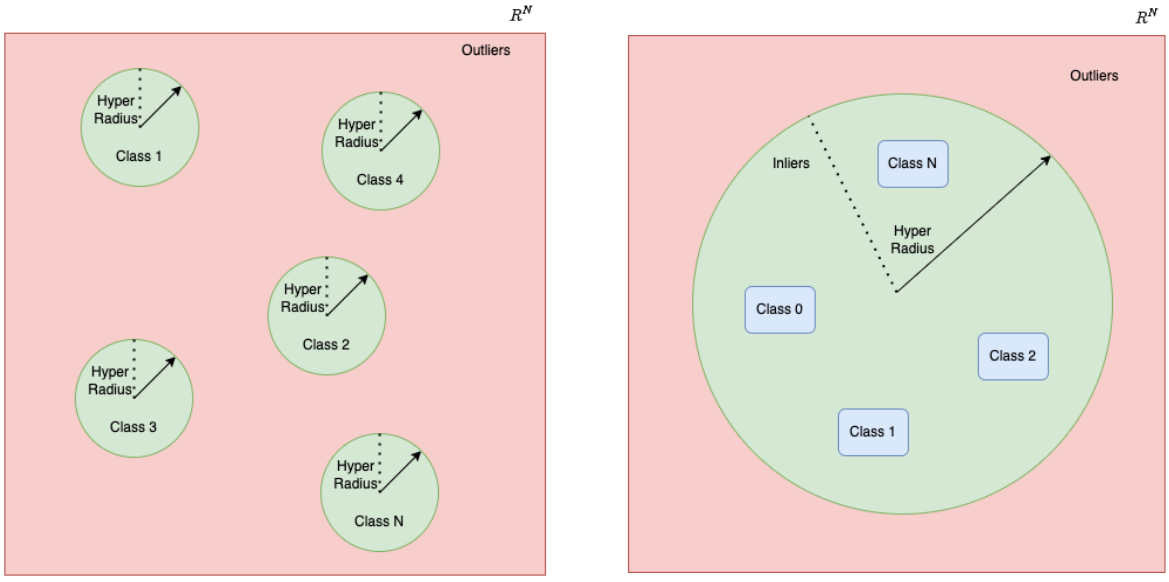

Figure 3: Left. (Proposed)The radius of each cluster is different. (Ruff et.al [6])Right. For every class, there is only one cluster

3.2 Pseudocode of the extended DOC training and testing

The training and testing pseudocodes are shown in Algorithm 1 and Algorithm 2, respectively. The optimization process is now updating the radius for each cluster using the samples for the respective data points. Here in the experiments, ν is defaulted as 0.1. We update the radius after warmup epochs similarly as done by the authors in the original paper.

During testing, if the data point scores are positive, then the point is considered an outlier.

Algorithm 1: PyTorch-style pseudocode of training stage 3 - extended DOC training

```
# Inputs:  c(shape=(num_clusters, rep_dims)), radius
 R(size=(num_clusters,)), data_loader, epochs, net, warm_n_epochs
for i in range(epochs):
   for data in enumerate(data_loader):
      images, pseudo_labels = data
      outputs = net(images)
      dist = torch.sum((outputs - c[pseudo_labels]) ** 2, dim=1)
      loss = 0
      for idx in range(self.c.shape[0]):
         scores = dist[pseudo_labels == idx] - R[idx] ** 2
         loss += torch.sum(R[idx] ** 2 + (1 / nu) *
          torch.mean(torch.max(torch.zeros_like(scores), scores)))
         # This is the equation of the theorem 3.  Minimizing the
          radius for each cluster and taking the penalty.
      loss /= self.c.shape[0]
      loss.backward()
      optimizer.step()
   if i>= warm_n_epochs:  # Update the radius using line search
      for idx in range(c.shape[0]):
         quantile =
          torch.tensor(np.quantile(np.sqrt(dist[pseudo_labels ==
          idx]), 1 - nu))
         # Updating the radius using the quantile based approach.
          Taking the 90th quantile by default.
         R[idx] = quantile
```

Algorithm 2: PyTorch-style pseudocode of inference - extended DOC testing

```
# Inputs:  cluster_centroids c(shape=(num_clusters, rep_dims)), radius
 R(size=(num_clusters,)), test_data_loader
anomaly_idxs = list()
for data in enumerate(test_data_loader):
   images, _ = data
   outputs = net(images)
   dist, idxs = ((outputs - c[:, None]) ** 2).sum(dim=2).min(dim=0)
   scores = dist - R[idxs] ** 2
anomaly_idxs.append(np.where(scores <= 0).list())
```

4 Implementation

All the experiments were done on the MNIST and CIFAR10 datasets. Each of the datasets has 10 different classes, and during the experiments randomly group of 2, 3, and 4 classes were combined to act as normal classes in MNIST, and groups of 2 and 3 classes were randomly combined to act as normal classes in the CIFAR10 dataset.

Digit	MNIST	
	Train	Test
0	5,923	980
1	6,742	1,135
2	5,958	1,032
3	6,131	1,010
4	5,842	982
5	5,421	892
6	5,918	958
7	6,265	1,028
8	5,851	974
9	5,949	1,009
Total	60,000	10,000

Figure 4: Distribution of MNIST dataset

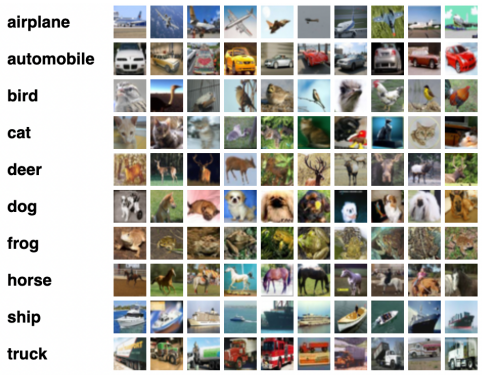

Figure 5: Classes in CIFAR10 dataset

Both the datasets were trained using the LeNet like CNN architectures, where each convolutional module consists of a convolutional layer followed by leaky ReLU activations and 2 × 2 max-pooling. On MNIST, we use a CNN with two modules, 8×(5×5×1)-filters followed by 4×(5×5×1)- filters, and a final dense layer of 32 units. On CIFAR-10, we use a CNN with three modules, 32 × (5 × 5 × 3)-filters, 64×(5×5×3)-filters, and 128×(5×5×3)-filters, followed by a final dense layer of 128 units. We use a batch size of 200 and set the weight decay hyperparameter to 10−6 [**Quoted from [5]**]

Three-phase training was employed, and the hyperparameters in training stage 1 and 3 were taken exactly the same to compare the direct results of the training. In both stages, the model was trained for 150 epochs with a batch size of 200. The initial learning rate η 10e-4 was gradually decreased using multi-step lr scheduler at every 50 epochs with a step size of 0.1. The leaky ReLU activation function was used with a leakiness of 0.1.

5 Results

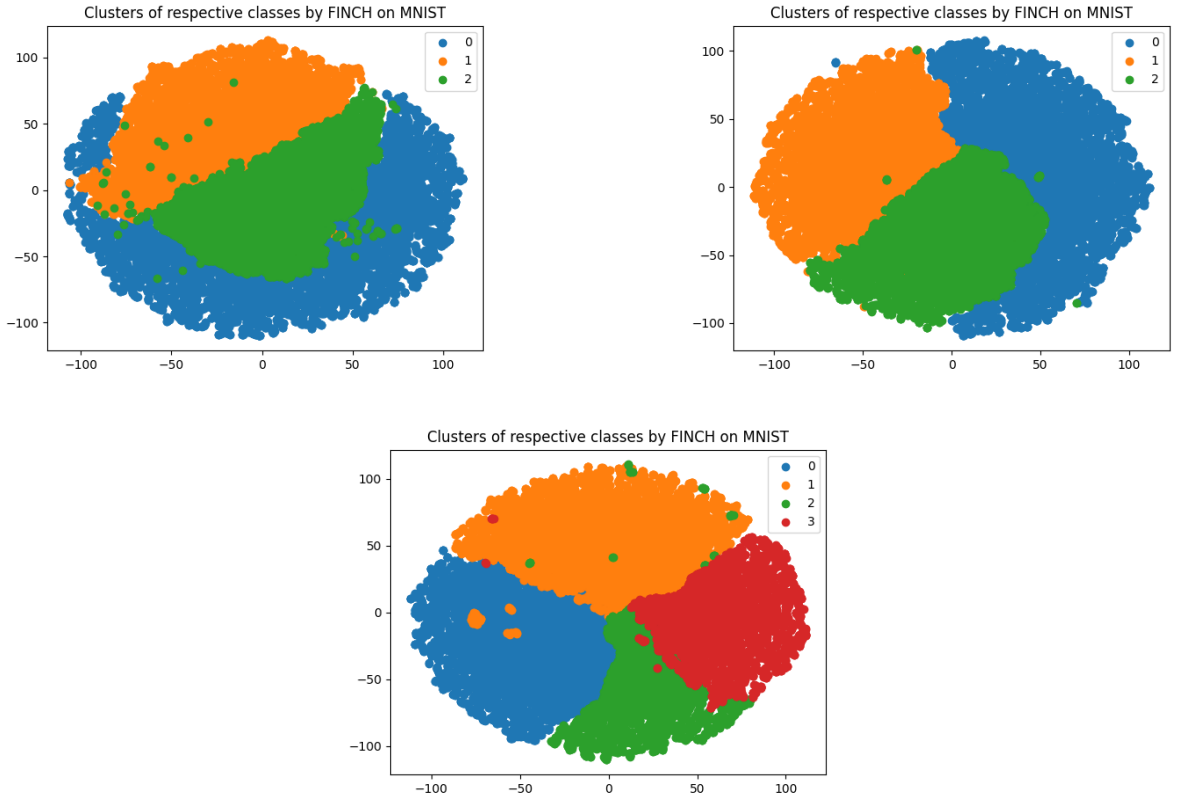

Figure 6: (**Left.** Original Labels, **Middle.** Clustering results)Results of the clustering using **FINCH** [6] algorithm on random three classes of MNIST dataset. **Right.** Overclustering by FINCH algorithm - It clusters the 3 groups of classes into 4 different assignments. We can see that the results of the original and clustering results are very similar to each other(Visualized using TSNE).

As shown in Fig.6 , FINCH[6] was able to cluster the embeddings exactly into 3 different clusters as compared to the ground truth. But sometimes, based on the embedding/pretraining stage results, the algorithm over-clusters or under-clusters the data. This may reduce the number of data points in each cluster which can haywire the training. Also, these pseudo labels can be taken into consideration even if they are weakly supervised, as the number of anomalous examples will be very less in the training data(considering the practical

scenario) and so the radius learning on an average will be biased towards the assignments of the true/normal classes.

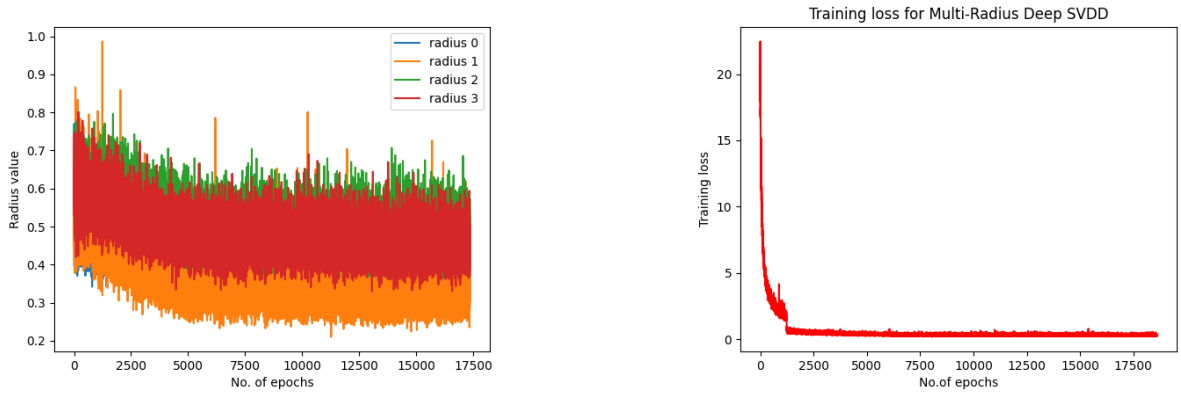

Figure 7: (**Left.** Radius convergence for different clusters on four random classes of MNIST dataset. **Right** Training loss convergence with the number of iterations.

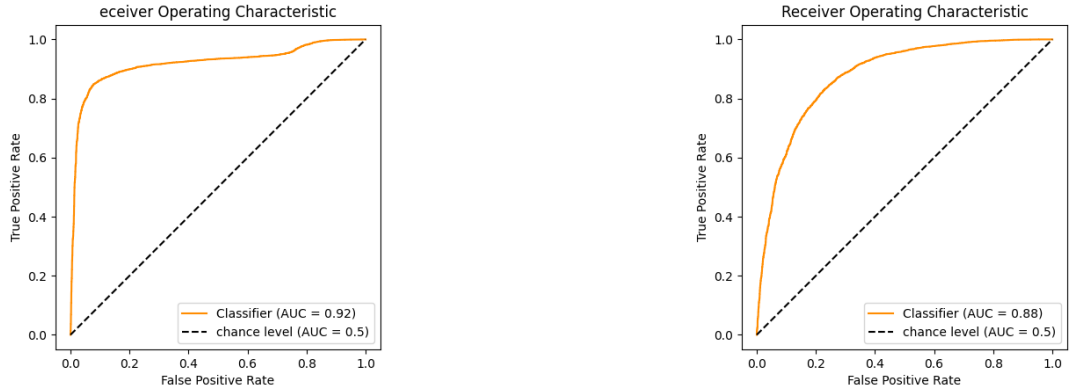

Figure 8: Results of the AUC curves. The class [0, 1, 3] in MNIST are taken as normal classes and the rest as outliers.**Left.** Multi-radius Deep SVDD **Right** Original algorithm by Ruff.et al[5]

As seen in the observations and **Fig. 8**, there was a huge dip in the AUC if we added more normal classes during the training. Over-clustering sometimes helped us achieve good results as each class was segregated into different categories in the representation space. Therefore, more fine-grained details can be captured by each cluster's hypersphere. But sometimes, due to fewer candidates per cluster(in over-clustering cases), the radius values were very large, which decreased the AUC drastically. More stable clustering algorithms or taking into account a pseudo-clustering objective function during the training can help us to achieve better results.

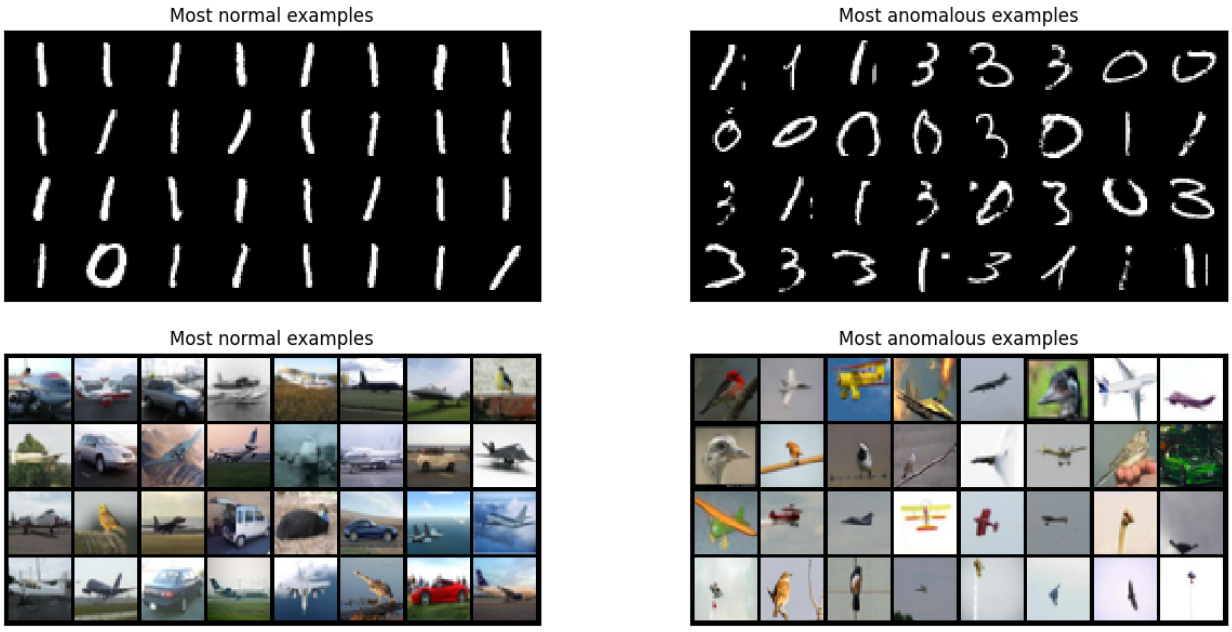

Figure 9: Intra-level normal and outlier classes - Images that are found to be inliers and outliers in the training data. **Above.** MNIST normal and outlier classes in the training data. **Below.** CIFAR10 normal and outlier classes in the training data

MNIST Dataset		
Number of groups	Original Deep SVDD	Multi-Radius SVDD
2	94.34	**96.32**
3	91.22	**95.31**
4	89.33	**93.36**
CIFAR10 Dataset		
Number of groups	Original Deep SVDD	Multi-Radius SVDD
2	57	**60.2**
3	54	**59.31**
4	-	-

Table 1: AUC Results of the Deep SVDD vs Multi Radius SVDD on MNIST and CIFAR10 datasets. We can see that the AUC increases drastically when more classes are added to the inliers.

6 Conclusion/Future Work/Improvements

We employ a multi-radius learning strategy in the Deep SVDD[5] algorithm, which helped us drastically improve the anomaly detection results in the cases when some of the classes are mixed up during the training. We leverage the FINCH [6] algorithm to generate pseudo-labels and learn the compactness of the hypersphere. Below given points can be taken into consideration for future work.

- **FINCH**[6] algorithm is very sensitive to the pretraining stage, which sometimes over-clusters the embedding space. This reduces the number of examples in the cluster and makes the radius-learning unstable.
- Instead of using the standalone clustering algorithm, we can add a surrogate clustering loss, such as contrastive loss and joint embedding loss functions, to further improve the results and remove the dependency of stage 2 of training.
- The method is susceptible to mode-collapse, where the DOC objective makes the clusters very compact. This makes the radius very small; mostly, all the normal examples are classified as anomalies.
- The paper by Goyal et.al[2] - Deep Robust One-Class Classification tries to solve the mode-collapse issue using adversarial examples and loss functions. We can leverage the idea from this paper to make this method more robust towards mode-collapse and clustering sensitivity.
- Instead of using simple auto-encoders, we can use Variational Auto-encoders[4], which can help us in getting more important representations of an image.

References

[1] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014.

[2] Sachin Goyal et al. "DROCC: Deep Robust One-Class Classification". In: *CoRR* abs/2002.12718 (2020). arXiv: 2002.12718. URL: https://arxiv.org/abs/2002.12718.

[3] G.E. Hinton and R.R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science (New York, N.Y.)* 313 (Aug. 2006), pp. 504–7. DOI: 10.1126/science.1127647.

[4] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2013. DOI: 10.48550/ARXIV.1312.6114. URL: https://arxiv.org/abs/1312.6114.

[5] Lukas Ruff et al. "Deep One-Class Classification". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 4393–4402. URL: https://proceedings.mlr.press/v80/ruff18a.html.

[6] M. Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. "Efficient Parameter-free Clustering Using First Neighbor Relations". In: *CoRR* abs/1902.11266 (2019). arXiv: 1902.11266. URL: http://arxiv.org/abs/1902.11266.

[7] Thomas Schlegl et al. "Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery". In: *CoRR* abs/1703.05921 (2017). arXiv: 1703.05921. URL: http://arxiv.org/abs/1703.05921.

[8] Bernhard Schölkopf et al. "Estimating Support of a High-Dimensional Distribution". In: *Neural Computation* 13 (July 2001), pp. 1443–1471. DOI: 10.1162/089976601750264965.

[9] David Tax and Robert Duin. "Support Vector Data Description". In: *Machine Learning* 54 (Jan. 2004), pp. 45–66. DOI: 10.1023/B:MACH.0000008084.60811.49.