UNSUPERVISED ANOMALY DETECTION VIA DEEP METRIC LEARNING WITH END-TO-END OPTIMIZATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

By
Selim Fırat Yılmaz
July 2021

Unsupervised Anomaly Detection via Deep Metric Learning with End-to-End Optimization
By Selim Fırat Yılmaz
July 2021

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Süleyman Serdar Kozat(Advisor)

Hamdi Dibeklioğlu

Ramazan Gökberk Cinbiş

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

UNSUPERVISED ANOMALY DETECTION VIA DEEP METRIC LEARNING WITH END-TO-END OPTIMIZATION

Selim Fırat Yılmaz
M.S. in Electrical and Electronics Engineering
Advisor: Süleyman Serdar Kozat
July 2021

We investigate unsupervised anomaly detection for high-dimensional data and introduce a deep metric learning (DML) based framework. In particular, we learn a distance metric through a deep neural network. Through this metric, we project the data into the metric space that better separates the anomalies from the normal data and reduces the effect of the curse of dimensionality for high-dimensional data. We present a novel data distillation method through self-supervision to remedy the conventional practice of assuming all data as normal. We also employ the hard mining technique from the DML literature. We show these components improve the performance of our model. Through an extensive set of experiments on the 14 real-world datasets, our method demonstrates significant performance gains compared to the state-of-the-art unsupervised anomaly detection methods, e.g., an absolute improvement between 4.44% and 11.74% on the average over the 14 datasets. Furthermore, we share the source code of our method on Github to facilitate further research.

Keywords: Anomaly detection, unsupervised, one-class, deep metric learning, data distillation.

ÖZET

DERİN METRİK ÖĞRENMESİ İLE BAŞTAN SONA OPTİMİZE EDİLEBİLEN GÖZETİMSİZ ANOMALİ TESPİTİ

Selim Fırat Yılmaz
Elektrik ve Elektronik Mühendisliği, Yüksek Lisans
Tez Danışmanı: Süleyman Serdar Kozat
Temmuz 2021

Bu çalışmakta yüksek boyutlu veriler için denetimsiz anormallik tespitini araştırılmakta ve derin metrik öğrenme tabanlı bir yapı sunulmaktadır. Derin sinir ağı aracılığıyla bir mesafe metriği öğrenilmektedir. Bu metrik aracılığıyla veriler daha küçük boyutlu metrik alana yansıtılmaktadır. Bu sayede anormallikleri normal verilerden daha iyi ayrılmakta ve yüksek boyutlu veriler için çok boyutluluğun lanetinin etkisi azaltılmaktadır. Tüm verileri normal kabul eden geleneksel uygulama yerine kendi kendine denetim yoluyla veriyi anormalliklerden arındıran yeni bir veri saflaştırma yöntemi sunulmaktadır. Ayrıca DML literatüründeki sert madencilik tekniğini kullanılmaktadır. Bu bileşenlerin ayrı ayrı modelimizin performansını iyileştirdiği gösterilmektedir. Literatürde yaygın olarak kullanılan 14 farklı gerçek dünya veri kümesinde yapılan kapsamlı deneyler sayesinde yöntemimiz, literatürdeki popüler denetimsiz anormallik algılama yöntemlerine kıyasla önemli performans kazanımları ortaya koymakta ve bu 14 veri kümesinin ortalamasında %4.44 ile %11.74 arasında ortalama eğri altındaki alan metriğinde artış sağlamaktadır. Tüm bunların yanında başka araştırmalara öncülük etmek adına metodumuzun kaynak kodu Github'da herkese açık şekilde paylaşılmaktadır.

Anahtar sözcükler: Anomali tespiti, gözetimsiz, tek sınıflı, derin metrik öğrenmesi, veri saflaştırma.

Acknowledgement

First and foremost, I would like to thank my advisor, Prof. Süleyman Serdar Kozat for his great guidance, encouragement and continuous support throughout my undergraduate and M.S. studies during the last three years. I would also like to thank Prof. Hamdi Dibeklioğlu for his invaluable guidance on various research projects during the previous three years.

I would like to thank Prof. Ramazan Gökberk Cinbiş and Prof. Hamdi Dibeklioğlu for being in my thesis committee and their time.

I would like to express my gratitude to my mother Kadriye, my father Ömer and my brother Anıl for their unconditional support and encouragement throughout my life. I would also like to thank Simge for her encouragement and devotion during this whole adventure.

I would like to acknowledge that this study is supported by Vodafone Turkey within the framework of 5G and Beyond Joint Graduate Support Programme coordinated by Information and Communication Technologies Authority.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We study anomaly detection [1], which has been extensively studied over recent years due to its wide range of applications such as video surveillance [2], price manipulation detection [3], network intrusion detection [4] and data imputation [5]. Anomaly detection aims to find the patterns that do not conform to the expected behavior [1]. Anomaly detection is an essential problem due to its usefulness in providing critical and actionable information on various domains [1]. Particularly, we study the anomaly detection problem in the unsupervised setting, where no labels exist. In the anomaly detection problem, the "normal" data is generally assumed to conform to a model, which is often unknown. Then, the anomalies are defined as the instances that significantly deviate from the underlying model of the normal data, where "significant deviances" are hard to model in real life and usually application dependent [6]. In this sense, we introduce an algorithm to remedy such application dependent components and introduce a generic and widely applicable algorithm in different domains.

Accurately labeling anomalies under human supervision is costly and even impractical in many cases such as manually labeling anomalies in large social networks [6]. Moreover, anomalies generally are rare among the normal behavior of the data, thus, filtering the normal data out requires a vast amount of human effort [6]. Furthermore, new types of anomalies may arise, e.g., zero-day

attacks [4], which do not exist in the set of the labeled anomalies [1]. Accordingly, an unsupervised anomaly detection method is more broadly applicable compared to the semi-supervised and the supervised ones, however, more prone to increased false-alarm rates, which severally limits their direct application in real life problems [1].

Although significant development has been made in recent years [6], anomaly detection in unsupervised setting remains challenging, especially on higher dimensions. Moreover, the running time naturally increases as the number of the dimensions increase, which should be avoided since anomalous behavior should be determined in real time for quick action. To reduce the effect of the curse of dimensionality, various two-step approaches have been proposed [6]. In these methods, first, the data is converted into low dimensional latent vectors via random subspace selection, dimensionality reduction, or random projection [6]. Then, in the second step, a model is fitted to represent the normality in the lower-dimensional data, which is later used to score the anomalies. However, the initial dimensionality reduction step is independent of the latter model fitting step, which produces suboptimal performance since the essential information to detect the anomalies could be removed in the first stage.

To remedy those issues, we introduce a novel "end-to-end" anomaly detection method based on deep metric learning (DML) for the unsupervised setting. In particular, we project the data through a network into the metric space, in which we optimize the similarity-based loss between distilled training instances, where both components are optimized jointly, unlike the previous approaches [6]. We derive an anomaly scoring function with $\Theta(1)$[1] time and memory complexity. We introduce a self-supervised data distillation method, which naturally fits our derived scoring function. Specifically, throughout the epochs, we select the instances that are labeled as likely to be normal by our method and optimize the model only using these instances. We also introduce hard normal mining method for our unsupervised framework [7]. Through our hard normal mining method, we only learn from the instances that are the farthest to the metric-space center

[1] $\Theta(w(n))$ denotes the set of all $r(n)$, where $a_1 w(n) \leq r(n) \leq a_2 w(n)$, $\forall n > n_0$ for $n \in \mathbb{Z}^+$ such that there exist positive integers a_1, a_2, and n_0.

of our distilled data, i.e., the most challenging instances, which further improves the model by avoiding overfitting to the easiest instances. We precisely define our methodology so that further instance mining methods and loss functions can be readily adapted from the DML literature using our method. Moreover, our method can be readily adapted to other data structures, e.g., convolutional neural networks (CNNs) [8] can be used for anomaly detection on images, and recurrent neural networks (RNNs) [9] can be used for anomaly detection on time series. We provide such adaptions as remarks in this thesis.

1.1 Related Work

Deep learning is a type of representation learning, which allows to learn data representations through multiple layers [10–12]. Learning the network parameters within such layered architecture permits to learn multiple abstractions of data with different complexity. Moreover, one can tune the network capacity by using different number of layers or neurons. This layered representation is also appropriate for data types that have hierarchical nature such as text and image [13,14]. Our method can be adapted to other data types besides multivariate vectors such as texts through LSTMs or images through CNNs.

Deep neural networks have been extensively studied in different areas, including anomaly detection due to their ability to model complex patterns [11]. To benefit from this capability, metric learning is extended to the deep learning framework in many forms such as the Siamese network [15]. The Contrastive Loss [16] is also proposed, which pulls the instances of the same class and pushes the instances of different classes in the metric space. This loss function later extended in several studies and achieved outstanding results in variety of tasks such as biometric recognition [17], image classification [18], object recognition [19], and image verification [20]. Promoting hard pairs in training, i.e., hard mining, improves the discrimination capability of the DML model in the supervised setting [7, 21]. Accordingly, we introduce the hard normal mining method into our model for the unsupervised anomaly detection problem. Although the DML based

methods received great attention in such supervised tasks, they have not received enough attention in the unsupervised anomaly detection literature. Masana et al. [22] employ DML for anomaly detection; however, their approach only works for the supervised setting, i.e., require labeled instances. Du and Zhang [23] employs conventional Mahalanobis metric learning. To the best of our knowledge, we are the first in the literature to introduce an unsupervised anomaly detection method based on the DML, which is optimized end-to-end.

Anomaly detection has been applied to many domains in terms of data type including natural language, time series, image, video, categorical, and multivariate data [6]. Anomaly detection methods can be categorized into three in terms of the availability of the labels. Supervised anomaly detection methods are only applicable to the domains where labeled training data are available. However, due to the rarity of anomalies in the training data, it is not easy to obtain labeled data [24]. Binary or multi-class classification methods can be used to tackle this problem, yet, they perform sub-optimal because of the class-imbalance problem [24]. Semi-supervised anomaly detection only requires normal training data. Semi-supervised anomaly detectors are more popular compared to supervised anomaly detection since many applications exist where normal data can be easily obtained such as intrusion detection [6]. Unsupervised anomaly detection methods do not require any labeled data and are generally applied to training data with anomalies. These methods utilize intrinsic properties of the data [24]. We introduce an unsupervised anomaly detection method that is trained end-to-end. Thus our method does not require any labels.

In the classical anomaly detection literature, various methods have also been proposed that incorporate support vector machines (SVM) [25], density estimation [26], information theory [27], clustering [28], and nearest-neighbor graphs [29]. Some hybrid approaches [30] employ deep networks such as CNN to extract features and feed these features into classical anomaly detectors.

Deep learning-based unsupervised anomaly detection methods almost exclusively aim to minimize the reconstruction error of autoencoder [6, 31], or the

One-Class SVM (OC-SVM) formulation [2, 32, 33]. Autoencoder (AE) based approaches require the encoder and the decoder networks. The encoder network projects the input into latent space. The decoder network tries to reconstruct the original input from the latent space. The decoder and the encoder structure is often symmetric. Our model requires only the encoder network and directly optimizes it end-to-end. Hence, our model requires nearly half of the parameters of the AE to project into the latent (metric) space, which is advantageous since the training accelerates as the number of parameters that need to be learned reduces [6]. Moreover, we distill the data from possible outliers using our model, whereas AE and OC-SVM based approaches treat outliers as normal instances, which causes overfitting since neural networks are sensitive to the outliers [6].

In this work, we use the DML in unsupervised anomaly detection domain. The common practice in the unsupervised anomaly detection is to assume all the training data as normal [7]. However, this assumption does not always hold in practice since the outliers occur within the typical process in many systems. Thus, such methods suffer in performance when the training data is polluted with the anomalies [1] (see Section 3.5). To reduce the effect of the polluted data, we introduce a data distillation method, which cleans the training data from outliers before each epoch through self-supervision. We also apply hard normal mining to prevent overfitting, which is a common issue in neural network based anomaly detection methods [6].

1.2 Contributions

Our contributions are as follows.

1. For the first time in the literature, we introduce a DML based unsupervised anomaly detection method, where the method is optimized "end-to-end" in a fully unsupervised setting.

2. We derive a center distance based anomaly scoring function with $\Theta(1)$ time

and memory complexity compared to the dissimilarity based scoring function of $\Theta(|\mathcal{M}|)$ time and memory complexity where $|\mathcal{M}|$ is the number of instances in the training set.

3. We introduce a novel self-supervision based data distillation method, which naturally fits into our prediction function with constant time complexity. This data distillation methodology is generic so that one can quickly adapt to any epoch-wise method such as deep autoencoders [6].

4. We also present an online hard normal mining algorithm that selects the most challenging instances in the distilled training data, which prevents the overfitting to the easiest instances.

5. Through an extensive set of experiments on 14 distinct real-world datasets, we demonstrate that our method significantly outperforms the state-of-the-art methods for unsupervised and one-class training settings. Our method achieves an absolute improvement between 4.44% and 11.74% on the average over 14 datasets that are widely used in the anomaly detection literature.

6. We perform an ablation study to demonstrate the performance gains achieved by the components of our method.

7. For reproducibility and to facilitate further research, the source code of our method is available on http://github.com/selimfirat/addml/.

1.3 Organization of the Thesis

The organization of the rest of this thesis is as follows.

- Chapter 2 - Novel Anomaly Detection Algorithm
  In this chapter, we first define unsupervised anomaly detection problem. We then introduce our deep metric learning based framework. Later, we

introduce the data distillation and online hard normal mining components. Lastly, we thoroughly describe the training details of our method.

- Chapter 3 - Simulations
  In this chapter, we first introduce evaluation methodology and experiment settings. We then present our ablation study. Later, we present the comparisons of our method with the baselines in three different settings. Lastly, we present our hyperparameter studies and empirically analyze the model collapse problem.

- Chapter 4 - Conclusion
  In this chapter, we conclude by providing concluding remarks.

Chapter 2

Novel Anomaly Detection Algorithm

In this chapter, we first formulate the unsupervised anomaly detection problem and describe the supervised deep metric learning framework. We then provide how it will be used in the semi-supervised and the unsupervised setting via novel losses and data distillation method. Lastly, we provide the details of our training method.

2.1 Problem Description

In this article, all vectors are column vectors and denoted by boldfaced lowercase letters. All matrices are denoted by boldfaced uppercase letters. We denote the multisets via $\{\cdot\}$. A multiset is a set that can have more than one identical members. Table 2.1 summarizes the notation and symbols used in this work.

The learner f observes training data $\mathcal{M} = \{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^n$, where n is the number of training data. The corresponding ground truth labels $\{y_i \in \{0, 1\}\}_{i=1}^n$ are not available during the training in the unsupervised training and only positive

labeled data are available in one class training. The label $y_t = 0$ represents a normal instance whereas the label $y_t = 1$ represents an anomalous instance. Naturally, we consider anomalous instances as not normal instances. We define the scoring function $\mathcal{S}$ for a test datum $\boldsymbol{x}_t \in \mathbb{R}^d$ with true label $y_t \in \{0, 1\}$ as

$$\mathcal{S}(\boldsymbol{x}_t) = p(y_t = 1 \mid \boldsymbol{x}_t).$$

The $\mathcal{S}$ assigns a score representing the degree of outlierness of each instance. Through the assigned scores, we obtain the list of instances ranked by the degree of outlierness. In the ideal scenario, all anomalous instances in the list should have higher score than the normal instances. Through this ranked list, an analyst can choose a domain-specific cutoff threshold τ to output the most appropriate anomalies [1]. We construct a binary decision function $\hat{y}_t \in \{0, 1\}$ via thresholding the scoring function $\mathcal{S}$ by $\tau \in \mathbb{R}$ to make a decision, i.e.,

$$\hat{y}_t = \begin{cases} 1 \text{ (Anomaly)}, & \text{if } \mathcal{S}(\boldsymbol{x}_t) > \tau \\ 0 \text{ (Normal)}, & \text{if } \mathcal{S}(\boldsymbol{x}_t) \leq \tau. \end{cases} \tag{2.1}$$

Obtaining the threshold τ is out-of-scope of this work. One can obtain the threshold through evaluating performance on all possible thresholds on labeled development set [34–36], jointly optimizing the threshold with the training data via supervised techniques [37,38], self-supervised learning of threshold [37], or via unsupervised techniques [39, 40]. The threshold selection can also be considered as a design choice that changes the sensitivity of the system in the unsupervised setting [41]. Thus, we assume a domain-specific threshold τ is chosen by an analyst.

Now, given the learner f, the scoring function $\mathcal{S}$, the threshold τ, and the training data $\mathcal{M}$, we decide whether the test instance $\boldsymbol{x}_t \in \mathbb{R}^d$ is anomalous or not via $\hat{y}_t$. In the rest of this chapter, we introduce our method by defining the f and the $\mathcal{S}$.

Generic Notations:	
$\boldsymbol{a}$	vector
$\boldsymbol{A}$	matrix
$\{\cdot\}$	multiset
$\|\|\cdot\|\|_2$	euclidean norm of $\cdot$
$\|\cdot\|$	cardinality of $\cdot$
Constants, variables and functions:	
τ	decision threshold
$\mathcal{S}$	scoring function
$\boldsymbol{x}_i$	input vector of the i^{th} instance.
y_t	true label of $\boldsymbol{x}_t$
$\hat{y}_t$	binary decision function for $\boldsymbol{x}_t$
$\mathcal{M}$	training data
$\mathcal{V}$	validation data
$\mathcal{D}_{\boldsymbol{M}}$	Mahalanobis distance
f	neural network model
$\mathcal{D}_f$	euclidean distance in the metric space
$\mathcal{I}^-$	pairs of different classes
$\mathcal{I}^+$	pairs of the same class
$\mathcal{L}_{\text{supervised}}$	constrastive loss
$\mathcal{L}_{\text{instance}}$	instance-closeness loss
$\mathcal{S}_{\text{dissimilarity}}$	instance-closeness (dissimilarity) based scoring function
$\mathcal{L}_{\text{center}}$	center-closeness loss
$\mathcal{S}_{\text{center}}$	center-closeness based scoring function
$\boldsymbol{\mu}$	center of the training instances in the metric space
f_{best}	best model found during training
$\mathcal{R}$	retrieval set
ρ^n	ratio of instances in the normal enclosure region
τ^n	data distillation ratio
$\mathcal{P}$	distilled training data
ρ^h	normal mining ratio
τ^h	online hard normal mining threshold
λ	weight decay hyperparameter
e^v	maximum number of epochs without validation improvement
b	mini-batch size

Table 2.1: Summary of symbols used in this work.

2.2 Conventional Metric Learning

In this section, we describe the traditional Mahalanobis metric learning. Traditional distance metric learning-based methods learn a symmetric positive semidefinite matrix $\boldsymbol{M} \in \mathbb{R}^{d \times d}$ to compute the Mahalanobis distance $\mathcal{D}_{\boldsymbol{M}}$ as [42]:

$$\mathcal{D}_{\boldsymbol{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{M} (\boldsymbol{x}_i - \boldsymbol{x}_J)},$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $\boldsymbol{x}_j \in \mathbb{R}^d$. We can decompose $\boldsymbol{M} = \boldsymbol{L}^T \boldsymbol{L}$ for $\boldsymbol{L} \in \mathbb{R}^{q \times d}$ and $q \leq d$ since $\boldsymbol{M}$ is a positive semidefinite matrix, we have:

$$\begin{aligned} \mathcal{D}_{\boldsymbol{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{L}^T \boldsymbol{L} (\boldsymbol{x}_i - \boldsymbol{x}_j)} \\ &= \sqrt{(\boldsymbol{L}\boldsymbol{x}_i - \boldsymbol{L}\boldsymbol{x}_j)^T (\boldsymbol{L}\boldsymbol{x}_i - \boldsymbol{L}\boldsymbol{x}_j)} \\ &= ||\boldsymbol{L}\boldsymbol{x}_i - \boldsymbol{L}\boldsymbol{x}_j||_2 . \end{aligned}$$

Instead of learning the $\boldsymbol{M}$ with a positive semidefiniteness constraint, we can directly optimize the unconstrained $g(\boldsymbol{x}) = \boldsymbol{L}\boldsymbol{x}$. The function g is a linear transformation, which projects $\boldsymbol{x}$ into a (possibly) lower dimensional subspace. Using the linear transformation $g(\boldsymbol{x}) = \boldsymbol{L}\boldsymbol{x}$, we obtain

$$\mathcal{D}_{\boldsymbol{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = ||g(\boldsymbol{x}_i) - g(\boldsymbol{x}_j)||_2 .$$

In the following section, we introduce a deep learning based extension for the metric learning.

2.3 Deep Metric Learning (DML)

In this section, we show the transition from the Mahalanobis metric learning to the DML framework. Note that g cannot model the nonlinear relations in the data since it only multiplies the input with $\boldsymbol{L}$. We extend the linear mapping g to the nonlinear multi layer neural network by constructing a function f due to

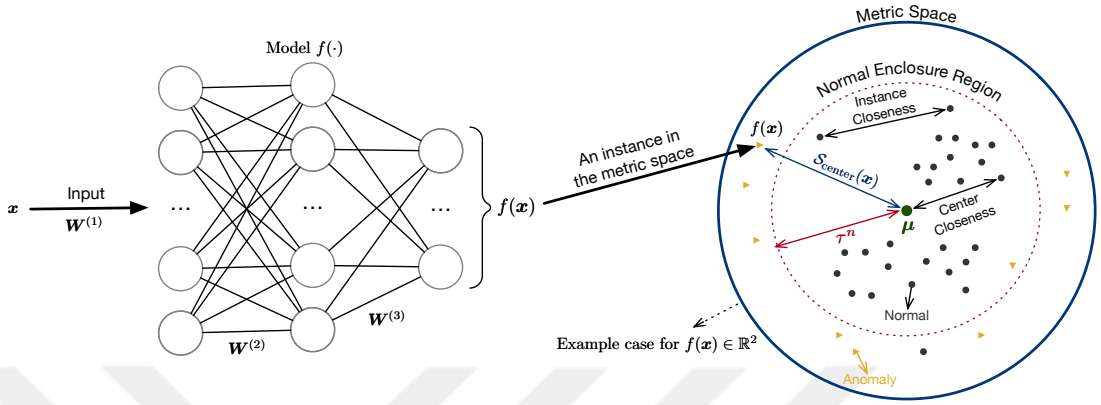

Figure 2.1: The figure illustrates the introduced method and the terms used. All the model parameters are jointly optimized. As the model $f(\cdot)$, one can use different approaches described in the text as remarks. The figure is best viewed in color.

the superior modeling power of the neural networks [11]. We denote the number of layers in the neural network as u, and the number of neurons in each layer as $v^{(c)}$, where $c \in \{1, 2, ...u\}$. We also denote the weight as $\boldsymbol{W}^{(c)} \in \mathbb{R}^{v^{(c)} \times v^{(c-1)}}$, and the bias as $\boldsymbol{b}^{(c)} \in \mathbb{R}^{v^{(c)}}$ for each layer. Let $\boldsymbol{h}^{(c)} = \tanh(\boldsymbol{W}^{(c)}\boldsymbol{h}^{(c-1)} + \boldsymbol{b}^{(c)})$ for the layer c and $\boldsymbol{h}^{(0)} = \boldsymbol{x}$. Then, we construct the neural network $f : \mathbb{R}^d \to \mathbb{R}^w$ through its u^{th} layer as:

$$\begin{aligned} f(\boldsymbol{x}) &= \boldsymbol{h}^{(u)} \\ &= \tanh(\boldsymbol{W}^{(u)}\boldsymbol{h}^{(u-1)} + \boldsymbol{b}^{(u)}), \end{aligned}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is the input vector, u is the final layer and $\boldsymbol{h}^{(u)} \in \mathbb{R}^w$ is the final output.

As shown in the Fig. 2.1, the network f transforms the d dimensional input vector $\boldsymbol{x}$ into a w dimensional latent vector $f(\boldsymbol{x})$. Then, the distance measure between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ becomes the euclidean distance between the latent representations $\mathcal{D}_f(\boldsymbol{x}_i, \boldsymbol{x}_j)$, which can also be seen as the generalized Euclidean distance as:

$$\mathcal{D}_f(\boldsymbol{x}_i, \boldsymbol{x}_j) = ||f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)||_2^2 .$$

We employ the distance $\mathcal{D}_f(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as a dissimilarity measure. We point out that other distance measures such as cosine distance can be used in our framework instead of $\mathcal{D}_f$. We denote the set of the positive pairs $\mathcal{I}^+ \subset \mathbb{R}^d \times \mathbb{R}^d$, which contains only the instances of the same type and the set of the negative pairs $\mathcal{I}^- \subset \mathbb{R}^d \times \mathbb{R}^d$, which contains only the instances of the different type as the following:

$$\mathcal{I}^+ = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) \mid y_i = y_j\}$$
$$\mathcal{I}^- = \{(\boldsymbol{x}_j, \boldsymbol{x}_k) \mid y_j \neq y_k\}.$$

The Contrastive loss [16] for the supervised training $\mathcal{L}_{\text{supervised}}$ is given by

$$\begin{aligned}\mathcal{L}_{\text{supervised}} = &\frac{1}{|\mathcal{I}^+| + |\mathcal{I}^-|}[\sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{I}^+} ||f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)||_2^2 \\ &+ \sum_{(\boldsymbol{x}_j, \boldsymbol{x}_k) \in \mathcal{I}^-} \max(0, a - ||f(\boldsymbol{x}_j) - f(\boldsymbol{x}_k)||_2^2)],\end{aligned}$$

where $a > 0$ represents the margin between normal and anomalous instances. Note that the Contrastive loss [16] requires the true labels due to its supervised nature. In the next section, we show how the supervised deep metric learning can be used in the one-class setting.

2.4 Metric Space Learning via Instance Closeness Loss

Before introducing the unsupervised framework, we first define our framework for the one-class training setting, i.e., all of the training data are known to be normal. In this setting, all the training data is known to be normal, and thus no pairs exist in $\mathcal{I}^-$. Moreover, $\mathcal{I}^+$ contains all possible pairs in this case. Thus, the

positive pairs $\mathcal{I}^+$ and the negative pairs $\mathcal{I}^-$ become equivalent to

$$\mathcal{I}^+ = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) \mid y_i = 0 \,\text{and}\, y_j = 0\} \,(\text{normal pairs})$$
$$I^- = \varnothing(\text{empty}).$$

Since $\mathcal{I}^-$ is empty in one-class setting, minimizing the loss $\mathcal{L}_{\text{supervised}}$ is equivalent to minimizing the loss $\mathcal{L}_{\text{instance}}$, which is defined as:

$$\mathcal{L}_{\text{instance}} = \frac{1}{|\mathcal{M}|^2} \sum_{i=1}^{|\mathcal{M}|-1} \sum_{j=i+1}^{|\mathcal{M}|} ||f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)||_2^2 .$$

***Remark* 1.** One can directly apply $\mathcal{L}_{\text{instance}}$ to the unsupervised setting following the common practice of assuming that all the training data is normal, similar to the conventional methods [6]. However, this practice highly degrades the performance of many anomaly detection methods [1] (see Section 3.5).

In the following section, we introduce another loss function that we later use to derive a feasible scoring function for $\mathcal{L}_{\text{instance}}$.

2.5 Metric Space Learning via Center Closeness Loss

We develop the center closeness loss $\mathcal{L}_{\text{center}}$, which is equivalent to the $\mathcal{L}_{\text{instance}}$ for the batch training setting.

Lemma 1. $\mathcal{L}_{\text{instance}}$ *is equivalent to the following statement*

$$\frac{1}{|\mathcal{M}|} \sum_{i=1}^{|\mathcal{M}|} ||f(\boldsymbol{x}_i) - \boldsymbol{\mu}||_2^2 ,$$

where $\boldsymbol{\mu} = \frac{1}{|\mathcal{M}|} \sum_{k=1}^{|\mathcal{M}|} f(\boldsymbol{x}_k)$.

Proof of Lemma 1:

$$
\begin{aligned}
\mathcal{L}_{\text{instance}} &= \frac{1}{|\mathcal{M}|^2} \sum_{i=1}^{|\mathcal{M}|-1} \sum_{j=i+1}^{|\mathcal{M}|} ||f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)||_2^2 \\
&= \frac{1}{2|\mathcal{M}|^2} \sum_{i=1}^{|\mathcal{M}|} \sum_{j=1}^{|\mathcal{M}|} ||f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)||_2^2 \\
&= \frac{1}{2|\mathcal{M}|^2} \sum_{i=1}^{|\mathcal{M}|} \sum_{j=1}^{|\mathcal{M}|} ||f(\boldsymbol{x}_i) - \boldsymbol{\mu} + \boldsymbol{\mu} - f(\boldsymbol{x}_j)||_2^2 \\
&= \frac{1}{2|\mathcal{M}|^2} \sum_{i=1}^{|\mathcal{M}|} \sum_{j=1}^{|\mathcal{M}|} ||f(\boldsymbol{x}_i) - \boldsymbol{\mu}||_2^2 \\
&\quad + \frac{1}{2|\mathcal{M}|^2} \sum_{i=1}^{|\mathcal{M}|} \sum_{j=1}^{|\mathcal{M}|} ||f(\boldsymbol{x}_j) - \boldsymbol{\mu}||_2^2 \\
&\quad - \frac{1}{|\mathcal{M}|^2} \sum_{i=1}^{|\mathcal{M}|} \sum_{j=1}^{|\mathcal{M}|} (f(\boldsymbol{x}_i) - \boldsymbol{\mu})^T (f(\boldsymbol{x}_j) - \boldsymbol{\mu}) \\
&= \frac{1}{|\mathcal{M}|} \sum_{i=1}^{|\mathcal{M}|} ||f(\boldsymbol{x}_i) - \boldsymbol{\mu}||_2^2 \\
&\quad - \frac{2}{|\mathcal{M}|^2} \sum_{i=1}^{|\mathcal{M}|} (f(\boldsymbol{x}_i) - \boldsymbol{\mu})^T (\sum_{j=1}^{|\mathcal{M}|} (f(\boldsymbol{x}_j) - \boldsymbol{\mu})).
\end{aligned}
$$

Since the term at the end $\sum_{j=1}^{|\mathcal{M}|}(f(\boldsymbol{x}_j) - \boldsymbol{\mu}) = \sum_{j=1}^{|\mathcal{M}|} f(\boldsymbol{x}_j) - \sum_{k=1}^{|\mathcal{M}|} f(\boldsymbol{x}_k) = 0$, $\mathcal{L}_{\text{instance}}$ is equal to the following term

$$
\frac{1}{|\mathcal{M}|} \sum_{i=1}^{|\mathcal{M}|} ||f(\boldsymbol{x}_i) - \boldsymbol{\mu}||_2^2 ,
$$

where $\boldsymbol{\mu} = \frac{1}{|\mathcal{M}|} \sum_{k=1}^{|\mathcal{M}|} f(\boldsymbol{x}_k)$. This concludes the proof of Lemma 1. □

***Remark* 2.** Lemma 1 shows that minimizing the distance of instances to the mean of f for the batch is equivalent to the minimizing the similarity between the instances for the batch case. This equivalence explains that we can obtain a

compact distribution of the data in the metric space for the normal samples by minimizing the similarity between instances, i.e., by minimizing $\mathcal{L}_{\text{instance}}$.

To further analyze the result of the Lemma 1, we define the center closeness loss $\mathcal{L}_{\text{center}}$ as:

$$\mathcal{L}_{\text{center}} = \frac{1}{|\mathcal{M}|} \sum_{i=1}^{|\mathcal{M}|} ||f(\boldsymbol{x}_i) - \boldsymbol{\mu}||_2^2 \,,$$

where $\boldsymbol{\mu} = \frac{1}{|\mathcal{M}|} \sum_{k=1}^{|\mathcal{M}|} f(\boldsymbol{x}_k)$. In the next section, we derive scoring functions using the $\mathcal{L}_{\text{instance}}$ and the $\mathcal{L}_{\text{center}}$.

2.6 Anomaly Scoring Functions

In this section, we introduce two different scoring functions $\mathcal{S}_{\text{dissimilarity}}$ and $\mathcal{S}_{\text{center}}$ to score the outlierness of the instances. Recall that we define the decision function $\hat{y}_t$ in (2.1) via thresholding the scoring function where the scores greater than τ are labeled as anomalies by our framework. Thus, our goal is to obtain larger scores for the anomalies and smaller scores for the normal instances. Minimizing $\mathcal{L}_{\text{instance}}$ minimizes the following expectation for the normal instances:

$$\mathop{\mathbb{E}}_{\boldsymbol{x}_t \in \mathcal{M}} \left[\frac{1}{|\mathcal{M}|} \sum_{\boldsymbol{x}_j \in \mathcal{M}} ||f(\boldsymbol{x}_t) - f(\boldsymbol{x}_j)||_2^2 \mid y_t = 0 \right]. \tag{2.2}$$

Accordingly, we define the euclidean distance (dissimilarity) for all instances as the scoring function as:

$$\mathcal{S}_{\text{dissimilarity}}(\boldsymbol{x}_t) = \frac{1}{|\mathcal{M}|} \sum_{\boldsymbol{x}_j \in \mathcal{M}} ||f(\boldsymbol{x}_t) - f(\boldsymbol{x}_j)||_2^2 \,.$$

However, this scoring function requires to calculate the distance to all instances, i.e., $\Theta(|\mathcal{M}|)$ distance calculations for scoring an instance. The number of training instances are usually too large for most applications such as intrusion detection [1]

and the time complexity of the $\mathcal{S}_{\text{dissimilarity}}$ grows with the training data size. Thus, employing $\mathcal{S}_{\text{dissimilarity}}$ as the scoring function is time-consuming and usually impractical. Since we show that $\mathcal{L}_{\text{instance}}$ is equivalent to the $\mathcal{L}_{\text{center}}$ via Lemma 1, hence both $\mathcal{L}_{\text{instance}}$ and $\mathcal{L}_{\text{center}}$ minimize the following expectation for the normal instances:

$$\mathop{\mathbb{E}}_{\boldsymbol{x}_t \in \mathcal{M}} \left[||f(\boldsymbol{x}_t) - \boldsymbol{\mu}||_2^2 \mid y_t = 0 \right], \tag{2.3}$$

where $\boldsymbol{\mu} = \frac{1}{|\mathcal{M}|} \sum_{k=1}^{|\mathcal{M}|} f(\boldsymbol{x}_k)$. Similar to the $\mathcal{S}_{\text{dissimilarity}}$, we define $\mathcal{S}_{\text{center}}$ as the euclidean distance to the center of the outputs of the model f as:

$$\mathcal{S}_{\text{center}}(\boldsymbol{x}_t) = ||f(\boldsymbol{x}_t) - \boldsymbol{\mu}||_2^2,$$

where $\boldsymbol{\mu} = \frac{1}{|\mathcal{M}|} \sum_{k=1}^{|\mathcal{M}|} f(\boldsymbol{x}_k)$ and f is the trained model. As shown in Fig. 2.1, anomalous instances appear farther to the $\boldsymbol{\mu}$ than the normal instances. Algorithm 1 summarizes the procedure for the $\mathcal{S}_{\text{dissimilarity}}$ as the *dissimilarity* method, and the $\mathcal{S}_{\text{center}}$ as the *center* method. We first build a retrieval set $\mathcal{R}$ by projecting all points into the metric space. We then calculate $\boldsymbol{\mu}$ using this retrieval set, and use $\boldsymbol{\mu}$ for center distance based scoring function. Using this retrieval set $\mathcal{R}$ and the center in the metric space $\boldsymbol{\mu}$, we infer for different

Algorithm 1: Dissimilarity & Center Based Scoring

Input: Model f_{best}, target instance $\boldsymbol{x}_t$, training data $\mathcal{M}$ (after validation split), *method* (name of the scoring function, i.e., *center* or *dissimilarity*)

Output: Score s of the target instance $\boldsymbol{x}$

Build Retrieval Set: (calculated only once before the inference time)

$\mathcal{R} = \{f_{\text{best}}(\boldsymbol{x}_i)\}_{\boldsymbol{x}_i \in \mathcal{M}}$

Calculate Center: (calculated only once before the inference time)

$\boldsymbol{\mu} = \frac{1}{|\mathcal{M}|} \sum_{\boldsymbol{r} \in \mathcal{R}} \boldsymbol{r}$

Calculate score:

if *method* = *dissimilarity* **then**

  $\boldsymbol{t} = f_{\text{best}}(\boldsymbol{x}_t)$

  $s = \sum_{\boldsymbol{r} \in \mathcal{R}} ||\boldsymbol{t} - \boldsymbol{r}||_2^2$

else if *method* = *center* **then**

  $s = ||f_{\text{best}}(\boldsymbol{x}_t) - \boldsymbol{\mu}||_2^2$

***Remark* 3.** Notice that $\mathcal{S}_{\text{center}}$ only requires to calculate the distance to the

center as shown in Fig. 2.1, thus, its time and memory complexities are $\Theta(1)$ unlike $\mathcal{S}_{\text{dissimilarity}}$ with $\Theta(|\mathcal{M}|)$ time and memory complexity. Since both of the $\mathcal{L}_{\text{instance}}$ and the $\mathcal{L}_{\text{center}}$ minimizes the expectation in (2.3), they can be used interchangeably with the $\mathcal{S}_{\text{center}}$.

In the next section, we introduce a self-supervised data distillation method that iteratively cleans the training data to obtain more reliable training data.

2.7 Self-Supervised Data Distillation

In this section, we introduce a data distillation method for the $\mathcal{L}_{\text{instance}}$ and the $\mathcal{L}_{\text{center}}$ via self-supervision that improves our method in the unsupervised setting.

Definition 1. SmallestK(A,K) returns the multiset of K elements with the lowest values in A.

Anomaly detection methods often assume nearly all the training data as normal and provide good performance when the assumption holds. However, their performance significantly deteriorates when this assumption is wrong [1]. To remedy this problem, we introduce a self-supervision based approach to distill the data between epochs while training. In this way, we reduce the anomaly rate of the training data throughout. After each epoch and in the beginning, we score all the training instances through $\mathcal{S}_{\text{center}}$. Using these scores, we distill the data by selecting ρ^n of the instances that are more likely to be normal than the rest of the training data according to our model. At the beginning of each epoch, we construct the set of distilled training data $\mathcal{P}$ as:

$$\mathcal{P} = \{\boldsymbol{x}_i \mid ||f(\boldsymbol{x}_i) - \boldsymbol{\mu}||_2 \leq \tau^n\}_{\boldsymbol{x}_i \in \mathcal{M}},$$

where

$$\boldsymbol{\mu} = \frac{1}{|\mathcal{M}|} \sum_{k=1}^{|\mathcal{M}|} f(\boldsymbol{x}_k)$$
$$\mathcal{F} = \{||f(\boldsymbol{x}_i) - \boldsymbol{\mu}||_2\}\boldsymbol{x}_{i \in \mathcal{M}}$$
$$\tau^n = \max(\text{SmallestK}(\mathcal{F}, K = \rho^n|\mathcal{M}|))$$

and $\rho^n \in (0, 1]$ is a hyperparameter that defines the ratio of the instances in $\mathcal{M}$, which we use to train our model for each epoch. As shown in Fig. 2.1, we select τ^n to enclose ρ^n of all instances in the training data. Then, we use those instances to build $\mathcal{P}$. Through training on $\mathcal{P}$ on a particular epoch, we build an enclosure region over the training data so that we consider the instances inside the region as normal during the epoch.

***Remark* 4.** Recall that $\mathcal{S}_{\text{center}}$ scores all the training data in a short time since its time complexity is $\Theta(1)$ Accordingly, the data distillation with $\rho^n < 1$ further reduces the running time at each epoch since it decreases the number of elements in the loss.

***Remark* 5.** Defining the hyperparameter ρ^n in terms of the ratio improves interpretability instead of directly defining τ^n as a hyperparameter. It also removes the requirement to choose a different τ^n manually for each dataset.

***Remark* 6.** We initialize the model f via uniform Glorot initialization [43]. Although the model is not trained in the beginning, it still can extract features from the data and projects the data similarly to the random projection. Random projection is applied before applying the anomaly detection methods and shown to be effective in anomaly detection tasks [6]. Our scoring functions are capable of scoring anomalies even with the randomly projected data. Thus, we apply the data distillation before the first epoch, too.

In the next section, we describe the construction of the mini-batches for the training.

2.8 Mini-batch Construction

We train our model through mini-batches of instances. We randomly select a small subset of size b without replacement from the training data $\mathcal{M}$. We repeat this process for iterations until we select all instances in $\mathcal{M}$ for each epoch. For the construction of the mini-batches, we use the distilled training data $\mathcal{P}$ after shuffling to facilitate the random sampling. Given that b is the mini-batch size and $\boldsymbol{\mu} = \frac{1}{|\mathcal{M}|}\sum_{k=1}^{|\mathcal{M}|} f(\boldsymbol{x}_k)$ is calculated at the beginning of each epoch, we construct the m^{th} mini-batch $\mathcal{B}^{(m)}$ of size b, and the corresponding set of distances $\mathcal{D}^{(m)}$ for the instance closeness loss as:

$$\mathcal{B}^{(m)} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) \mid i < j < mb\}_{i=(m-1)b+1}^{mb}$$
$$\mathcal{D}^{(m)} = \{||f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)||_2\}_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{B}^{(m)}}.$$

Similarly for the center closeness loss, we define the mini-batch $\mathcal{B}^{(m)}$ and the corresponding set of distances $\mathcal{D}^{(m)}$ as:

$$\mathcal{B}^{(m)} = \{\boldsymbol{x}_i\}_{i=(m-1)b+1}^{mb}$$
$$\mathcal{D}^{(m)} = \{||f(\boldsymbol{x}_i) - \boldsymbol{\mu}||_2\}_{\boldsymbol{x}_i \in \mathcal{B}^{(m)}}.$$

In the following section, we present the hard normal mining method.

2.9 Online Hard Normal Mining

In this section, we introduce the hard normal mining method for our framework that selects the specific instances for training, which are more likely to improve the model [7].

The idea behind hard mining is to select the samples that are challenging so that the model does not overfit to the easiest instances in the training data [7]. We perform the hard mining on the mini-batches, i.e., in online manner, to avoid the calculation of the distances for all pairs in the training data. Recall that

we define the normal enclosure region in Section 2.7. We construct mini-batches using the instances in the normal enclosure region, i.e., the distilled training data. Thus, the selected instances are less likely to be anomalies.

Definition 2. LargestK(A,K) returns the multiset of K elements with the largest values in A.

We only incorporate the hardest instances of ratio ρ^h in each mini-batch that are inside the normal enclosure region. For instance closeness loss, we define the hardest instances as the farthest pairs of instances in the enclosure region. For center closeness loss, we define the hardest instances as the farthest points to the center in the normal enclosure region. Through the set of distances in mini-batch $\mathcal{D}^{(m)}$, we select the set of hardest distances in the mini-batch m as:

$$\mathcal{Q}^{(m)} = \{r \mid r \geq \tau^h\}_{r \in \mathcal{D}^{(m)}},$$

where

$$\tau^h = \min(\text{LargestK}(\mathcal{D}^{(m)}, K = \rho^h |\mathcal{D}^{(m)}|))$$

and $\rho^h \in (0, 1]$ is a hyperparameter that defines the ratio of the distances in $\mathcal{D}^{(m)}$, which we use to train our model for each mini-batch.

***Remark* 7.** The online hard normal mining with $\rho^h < 1$ reduces the number of elements in the loss, therefore, it decreases the running time of our method.

We describe the training details of our model in the next section.

2.10 Training of the Model

In this section, we provide the details of the training and combine introduced methods described in the previous sections.

Using $\mathcal{Q}^{(m)}$, we unify the calculation of both losses and combine them with the weight decay regularization into $\mathcal{L}^{(m)}$ as:

$$\mathcal{L}^{(m)} = \frac{1}{|\mathcal{Q}^{(m)}|} \sum_{r \in \mathcal{Q}^{(m)}} r + \frac{\lambda}{2} \sum_{c=1}^{u} \left|\left| \boldsymbol{W}^{(c)} \right|\right|_F^2,$$

where u is the number of layers in the network, $\boldsymbol{W}^{(c)}$ is the weight matrix of the layer c, and λ is the weight decay regularization hyperparameter. After each mini-batch's forward pass, we update our model f. For each layer $c = \{1, ...u\}$, we update the layer weight matrix $\boldsymbol{W}^{(c)}$ and bias vector $\boldsymbol{b}^{(c)}$ using $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^{(c)}}$, and $\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}^{(c)}}$, respectively. We use Adam's algorithm [44] for parameter optimization of the model f. We calculate the center $\boldsymbol{\mu}$ at the beginning since it is not feasible to calculate it after each weight update.

We also adapt early stopping and the best model selection through the validation set $\mathcal{V}$. We select the best model f_{best} as the model with the lowest validation loss. If validation loss does not improve for e^v epochs, we stop the training and set f_{best} as the best model, i.e., the model with the lowest validation loss. Note that since our method is **unsupervised** and the losses do not require any labels, we do not use any labels from the validation set, too.

Algorithm 2 and 3 summarizes the training of the whole model for both center closeness and instance closeness losses, respectively.

In theory, our model can have all-zero-weights. This situation causes model collapse, i.e., $f(\boldsymbol{x}) = 0 \ \forall \boldsymbol{x}$. Moreover, when all instances are mapped to the same constant, loss again becomes zero for all instances, i.e., $f(\boldsymbol{x}) = c \ \forall \boldsymbol{x}$, where $c \in \mathbb{R}$. Therefore, several of decisions to construct our method makes model collapse unlikely to occur. First, we use tanh activation function instead of ReLU since ReLU outputs zero for all values less than zero whereas tanh has only one zero point. Secondly, we utilize early stopping when validation loss does not drop for e^v in a row. Thirdly, we limit number of maximum epochs to n. Fourthly, we use mini-batch training that introduces stochasticity to the model updates. Lastly, we utilize online hard normal mining, which hardens to collapse to zero.

Algorithm 2: Complete Training Procedure for *Instance Closeness* Loss

Input: Model f, training data $\mathcal{M}$, validation data $\mathcal{V}$
Output: Trained model f_{best} with the lowest validation error
Parameters: n (number of epochs), ρ^n (data distillation ratio), ρ^h (normal mining ratio), e^v (max # epochs without improvement), λ (weight decay), b (mini-batch size)

Initialize $\{\boldsymbol{W}^{(c)}\}_{c=1}^{u}$ via Uniform Glorot [43], $v = \infty$ (lowest validation loss), $z = e^v$ (number of epochs before early stopping)

Training:

for *epoch* $e = 1$ *to* n **do**
  if $z = 0$ **then**
    Early stopping
    break
  Data Distillation:
  $\boldsymbol{\mu} = \frac{1}{|\mathcal{M}|} \sum_{k=1}^{|\mathcal{M}|} f(\boldsymbol{x}_k)$
  $\mathcal{F} = \{||f(\boldsymbol{x}_i) - \boldsymbol{\mu}||_2\}_{\boldsymbol{x}_i \in \mathcal{M}}$
  $\tau^n = \max(\text{SmallestK}(\mathcal{F}, K = \rho^n |\mathcal{M}|))$
  $\mathcal{P} = \{\boldsymbol{x}_i \mid ||f(\boldsymbol{x}_i) - \boldsymbol{\mu}||_2 \leq \tau^n\}_{\boldsymbol{x}_i \in \mathcal{M}}$
  Construct Mini-batches:
  Shuffle $\mathcal{P}$ with random sampling
  for *mini-batch* $m = 1$ *to* $\left\lceil \frac{|\mathcal{P}|}{b} \right\rceil$ **do**
    $\mathcal{B}^{(m)} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) \mid i < j < mb\}_{i=(m-1)b+1}^{mb}$
  end
  for *mini-batch* $m = 1$ *to* $\left\lceil \frac{|\mathcal{P}|}{b} \right\rceil$ **do**
    Forward Pass:
    $\mathcal{D}^{(m)} = \{||f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)||_2\}_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{B}^{(m)}}$
    Online Hard Normal Mining:
    $\tau^h = \min(\text{LargestK}(\mathcal{D}^{(m)}, K = \rho^h |\mathcal{D}^{(m)}|))$
    $\mathcal{Q}^{(m)} = \{r \mid r \geq \tau^h\}_{r \in \mathcal{D}^{(m)}}$
    Calculate Loss:
    $\mathcal{L}^{(m)} = \frac{1}{|\mathcal{Q}^{(m)}|} \sum_{r \in \mathcal{Q}^{(m)}} r + \frac{\lambda}{2} \sum_{c=1}^{u} \left|\left| \boldsymbol{W}^{(c)} \right|\right|_F^2$
    Backward Pass: (Assume $\boldsymbol{\mu}$ is scalar)
    Update $\boldsymbol{W}^{(c)}$, $\boldsymbol{b}^{(c)}$ via backpropagation using $\frac{\partial \mathcal{L}^{(m)}}{\partial \boldsymbol{W}^{(c)}}$, $\frac{\partial \mathcal{L}^{(m)}}{\partial \boldsymbol{b}^{(c)}}$
    Validation:
    $\mathcal{D}_v = \{||f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)||_2\}_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{V} \times \mathcal{V}}$
    $\mathcal{L}_v = \sum_{r \in \mathcal{D}_v} r$
    if $\mathcal{L}_v < v$ **then**
      $v = \mathcal{L}_v$
      $z = e^v$
      $f_{\text{best}} = f$
    else
      $z = z - 1$
  end
end

Algorithm 3: Complete Training Procedure for *Center Closeness* Loss

Input: Model f, training data $\mathcal{M}$, validation data $\mathcal{V}$

Output: Trained model f_{best} with the lowest validation error

Parameters: n (number of epochs), ρ^n (data distillation ratio), ρ^h (normal mining ratio), e^v (max # epochs without improvement), λ (weight decay), b (mini-batch size)

Initialize $\{\boldsymbol{W}^{(c)}\}_{c=1}^{u}$ via Uniform Glorot [43], $v = \infty$ (lowest validation loss), $z = e^v$ (number of epochs before early stopping)

Training:

for *epoch* $e = 1$ *to* n **do**

  if $z = 0$ **then**

    Early stopping

    break

  Data Distillation:

  $\boldsymbol{\mu} = \frac{1}{|\mathcal{M}|} \sum_{k=1}^{|\mathcal{M}|} f(\boldsymbol{x}_k)$

  $\mathcal{F} = \{||f(\boldsymbol{x}_i) - \boldsymbol{\mu}||_2\}_{\boldsymbol{x}_i \in \mathcal{M}}$

  $\tau^n = \max(\text{SmallestK}(\mathcal{F}, K = \rho^n |\mathcal{M}|))$

  $\mathcal{P} = \{\boldsymbol{x}_i \mid ||f(\boldsymbol{x}_i) - \boldsymbol{\mu}||_2 \leq \tau^n\}_{\boldsymbol{x}_i \in \mathcal{M}}$

  Construct Mini-batches:

  Shuffle $\mathcal{P}$ with random sampling

  for *mini-batch* $m = 1$ *to* $\left\lceil \frac{|\mathcal{P}|}{b} \right\rceil$ **do**

    $\mathcal{B}^{(m)} = \{\boldsymbol{x}_i\}_{i=(m-1)b+1}^{mb}$

  end

  for *mini-batch* $m = 1$ *to* $\left\lceil \frac{|\mathcal{P}|}{b} \right\rceil$ **do**

    Forward Pass:

    $\mathcal{D}^{(m)} = \{||f(\boldsymbol{x}_i) - \boldsymbol{\mu}||_2\}_{\boldsymbol{x}_i \in \mathcal{B}^{(m)}}$

    Online Hard Normal Mining:

    $\tau^h = \min(\text{LargestK}(\mathcal{D}^{(m)}, K = \rho^h |\mathcal{D}^{(m)}|))$

    $\mathcal{Q}^{(m)} = \{r \mid r \geq \tau^h\}_{r \in \mathcal{D}^{(m)}}$

    Calculate Loss:

    $\mathcal{L}^{(m)} = \frac{1}{|\mathcal{Q}^{(m)}|} \sum_{r \in \mathcal{Q}^{(m)}} r + \frac{\lambda}{2} \sum_{c=1}^{u} \left|\left| \boldsymbol{W}^{(c)} \right|\right|_F^2$

    Backward Pass: (Assume $\boldsymbol{\mu}$ is scalar)

    Update $\boldsymbol{W}^{(c)}$, $\boldsymbol{b}^{(c)}$ via backpropagation using $\frac{\partial \mathcal{L}^{(m)}}{\partial \boldsymbol{W}^{(c)}}$, $\frac{\partial \mathcal{L}^{(m)}}{\partial \boldsymbol{b}^{(c)}}$

    Validation:

    $\mathcal{D}_v = \{||f(\boldsymbol{x}_i) - \boldsymbol{\mu}||_2\}_{\boldsymbol{x}_i \in \mathcal{V}}$

    $\mathcal{L}_v = \sum_{r \in \mathcal{D}_v} r$

    if $\mathcal{L}_v < v$ **then**

      $v = \mathcal{L}_v$

      $z = e^v$

      $f_{\text{best}} = f$

    else

      $z = z - 1$

  end

end

Hard normal mining selects ρ^h of the instances from each mini-batch that are farthest to the center, i.e., less likely to collapse to a single point. In Section 3.8, we empirically analyze the model collapse issue.

***Remark* 8.** We can readily extend our work to the CNNs [8] for anomaly detection on images since they are shown to be successful in image-related tasks. We are given a training set of RGB images $\{\boldsymbol{m}_i \in \mathbb{R}^{3\times s\times h}\}_{i=1}^{n}$, where s is the width, and h is the height. We first map the image to w dimensional latent vector via a convolutional network followed by flattening, i.e., $c(\cdot) : \mathbb{R}^{3\times s\times h} \rightarrow \mathbb{R}^d$, which may also contain a feed-forward extension at the end. Instead of the $f(\cdot)$ that processes only the vectors, we use $c(\boldsymbol{m})$ for $\boldsymbol{m} \in \mathbb{R}^{3\times s\times h}$.

***Remark* 9.** We can readily extend our framework to the RNNs [9] for anomaly detection on time series due to their ability to store the past information. The RNN outputs $\boldsymbol{h}_t = \tanh(\boldsymbol{U}\boldsymbol{x} + \boldsymbol{V}\boldsymbol{h}_{t-1})$, where t is the time step, w is the metric space size, weight $\boldsymbol{U} \in \mathbb{R}^{w\times d}$, weight $\boldsymbol{V} \in \mathbb{R}^{w\times w}$, $\boldsymbol{h}_t \in \mathbb{R}^w$ and $\boldsymbol{h}_0 \in \mathbb{R}^w$ is the initial hidden state vector. Instead of the $f(\cdot)$, we directly use $\boldsymbol{h}_t$. Notice that the network can be extended to the multiple layers and combined with other layers such as feed-forward layer as long as it outputs w dimensional latent vector.

In the following chapter, we demonstrate our experiments and their details.

Chapter 3

Simulations

In this section, we demonstrate the anomaly detection performance of our method on various real-world datasets and compare it with the state-of-the-art methods. We also analyze the behavior of the method with respect to its parameters and perform an ablation study.

3.1 Evaluation Methodology

In this section, we describe the evaluation metrics, the cross validation strategy and the training settings.

To compare the performance on the anomaly detection datasets, we use the Area Under Curve of the Receiver Operating Characteristics (AUC of the ROC) in [45–47], which is commonly used in the anomaly detection literature [6]. As in [48], we sample varying thresholds from the Receiver Operating Characteristics (ROC) [49, 50] curve of the each method. As we obtain l true positive rate and false positive rate pairs through the sampling of ROC, we calculate the AUC via

$$\sum_{i=1}^{l}(t_{i+1}+t_i)(f_{i+1}-f_i),$$

where t is the true positive rate, and f is the false positive rate. We compute the AUC score using the ground truth labels. We do not use any labels for the training of the models.

We apply the 3-fold stratified cross-validation. We split the data into 3-folds with equal anomaly rates and repeat the experiment by choosing one fold as the test set and the rest as the training set. We repeat this process for each of the three splits. To reduce the effect of randomness in our experiments, we repeat all cross-validation experiments three times with different random seeds. Hence, we report the mean score of the nine experiments for every method-dataset pair.

We report the scores for the three different settings for all experiments. For the *seen* setting, we train and test on the training set. For the *unseen* setting, we train on the training set and test on the test set. For the *one-class* setting, we manually remove anomalies from the training set and evaluate on the test set. In the following section, we describe the methods used in our comparisons and their implementation details.

3.2 Compared Methods and Implementation Details

We have introduced the Anomaly Detection with Deep Metric Learning (AD-DML) method in Section 2. We compare our method with Deep Autoencoder (DAE) [6], Isolation Forest (IF) [51], Robust Principal Component Classifier (PCC) [52], One-Class Support Vector Machine (OC-SVM) [25], and Histogram Based Outlier Scoring (HBOS) [53]. We select the hyperparameters as suggested in the particular papers [25, 51–53]. We use IF with 100 trees, and 256 samples (214 for *Glass* dataset) for each tree. We use OC-SVM with Gaussian kernel and $\nu = 0.5$. We use HBOS with 5 bins. For all experiments, we normalize the data via subtracting the mean of the training data and dividing by the standard deviation of the training data.

Hyperparameter	Value
Data distillation ratio ρ^n (seen and unseen settings)	$\frac{2}{3}$
Data distillation ratio ρ^n (one-class settings)	1
Hard normal mining ratio ρ^h	$\frac{1}{3}$
Metric space dimension w (number of neurons in the last layer)	64
Number of neurons in the hidden layer	128
Validation set ratio ρ^v	0.1
Maximum number of epochs n	50
Learning rate	10^{-3}
Weight decay λ	10^{-5}
Early stopping patience e^v	5

Table 3.1: Hyperparameters chosen for our method

We choose the hyperparameters listed in Table 3.1 for ADDML, which is our method, we use the same hyperparameters for all experiments except the *one-class* setting, where we use the normal ratio $\rho^n = 1$ due to the results of the ablation study in Section 3.4. We use $\rho^n = \frac{2}{3}$ for the *seen*, and the *unseen* settings. We use the instance closeness loss with the center distance scoring function $\mathcal{S}_{\text{center}}$ since the dissimilarity based scoring function $\mathcal{S}_{\text{dissimilarity}}$ is infeasible for large number of training instances. We use the hard mining ratio of $\rho^h = \frac{1}{3}$. We set number of neurons in the hidden layer to 128. We use metric space dimension (number of neurons in the last layer) $w = 64$, which is also the latent space dimension for the DAE. For the DAE and the ADDML, we use Adam [44] optimizer with the default learning rate 10^{-3} and the weight decay $\lambda = 10^{-5}$. We also use the $\rho^v = 0.1$ of the training set as a validation set. We train the models for $n = 50$ epochs with early stopping with $e^v = 5$ patience. We implement the ADDML via Pytorch [54]. In the next section, we specify the datasets and their statistics.

3.3 Datasets

Table 3.2 shows the datasets along with the number of dimensions, the number of instances and the percentage of the outliers in each dataset we use. These are the real life datasets widely used in the anomaly detection experiments in

Dataset	# Samples	# Dimensions	Outliers %
Gisette	3,850	4,971	9.09
Glass	214	9	4.21
Ionosphere	351	33	35.9
Isolet	4,886	617	7.96
Letter	1,600	32	6.25
Madelon	1,430	500	9.09
Magic-Telescope	13,283	10	7.16
Mnist	7,603	100	9.21
Musk	3,062	166	3.17
Pendigits	6,870	16	2.27
Satellite	6,435	36	31.64
Satimage-2	5,803	36	1.22
Shuttle	49,097	9	7.15
Vowels	1,456	12	3.43

Table 3.2: Statistics of the datasets used in the experiments.

the literature [51, 55] (see [56] and [57] for details). In the following section, we demonstrate the effect of the introduced components through an ablation study.

3.4 Ablation Study

In this section, we perform an ablation study to observe the effect of the data distillation and the hard mining components.

Table 3.3 shows the AUC scores on the *Letter* dataset when the components added one by one. We apply the evaluation methodology described in Section 3.1. We first evaluate the effect of the instance closeness, and center closeness losses without the data distillation and the hard mining components by setting $\rho^n = 1$ and $\rho^h = 1$. Then, we add the data distillation component via setting $\rho^n = \frac{2}{3}$. Lastly, we add the hard mining component by setting $\rho^h = \frac{1}{3}$.

When we include the data distillation to the instance closeness loss, the AUC increases by 3% (absolute) in the seen setting and 2% (absolute) in the unseen setting. The AUC also slightly increases in the seen and unseen settings

Components	Seen	Unseen	One-Class
Instance Closeness Loss	77.42	77.96	81.37
+ Data Distillation	80.51	80.17	80.83
+ Hard Normal Mining	**81.49**	**81.17**	**82.33**
Center Closeness Loss	79.80	79.63	78.87
+ Data Distillation	77.14	77.80	78.53
+ Hard Normal Mining	77.20	78.22	79.55

Table 3.3: Ablation study for the *Letter* dataset when the components are added gradually.

of the center closeness loss when we add the data distillation component. For both losses in the one-class setting, the performance decreases as expected when we add the data distillation component. Recall that the motivation behind the data distillation is to remove the anomalies from the training data so that the model would not overfit to those samples. However, there are no anomalies in the one-class setting. Thus, the performance reduces as the number of training instances at each epoch. The AUC slightly increases when we include the hard mining component for both losses and in all settings. The instance closeness loss is more accurate compared to the center closeness loss without the data distillation and the hard normal mining. Moreover, the instance closeness loss benefits significantly more by the data distillation and the hard normal mining components compared to the center closeness loss. Thus, in further experiments, we only analyze the instance closeness loss for fairness.

3.5 Comparison with the State-of-the-Art Methods

In this section, we compare our model with the state-of-the-art unsupervised anomaly detection methods from different paradigms. We use the evaluation and the cross-validation method described in Section 3.1. We use the hyperparameters described in Section 3.2.

Recall that we report the scores for the three different settings for all experiments. For the *seen* setting, we train and test on the training set reported in subsection 3.5.1. For the *unseen* setting reported in subsection 3.5.2, we train on the training set and test on the test set. For the *one-class* setting reported in subsection 3.5.3, we manually remove anomalies from the training set and evaluate on the test set. Finally, we outline our overall results in subsection 3.5.4.

3.5.1 Comparison in the *Seen* Setting

Table 3.4 shows the results in the seen setting over 9 experiments per model for each dataset. Among six different models on 14 different datasets, our method takes first place on 9 datasets, second place on 1 dataset, third place on 2 datasets, fourth place on 1 dataset and sixth place on 1 dataset. On average, our method improves the AUC in absolute between 4.71%-10.59% for the seen setting.

Dataset	DAE	IF	PCC	OC-SVM	HBOS	ADDML
Gisette	74.37	49.24	74.23	**75.02**	35.60	73.40
Glass	60.56	71.72	45.35	58.81	70.61	**73.57**
Ionosphere	84.99	85.65	79.24	85.14	65.46	**92.64**
Isolet	51.23	54.61	51.31	55.59	54.07	**59.33**
Letter	50.46	63.54	51.77	59.81	57.06	**81.49**
Madelon	50.47	51.93	50.50	50.48	**53.28**	51.57
Magic-Telescope	69.57	**77.85**	68.84	72.56	73.97	74.65
Mnist	85.85	79.78	84.88	85.00	65.68	**85.89**
Musk	**100.00**	99.92	99.99	**100.00**	**100.00**	**100.00**
Pendigits	93.69	**94.98**	92.49	93.16	92.51	86.91
Satellite	61.87	70.72	62.88	66.36	68.68	**80.88**
Satimage-2	97.77	99.41	97.36	99.67	97.74	**99.87**
Shuttle	99.00	**99.66**	99.06	99.17	98.19	99.14
Vowels	57.25	76.28	59.67	77.84	63.38	**85.21**
Average	74.08	76.81	72.68	77.04	71.16	**81.75**

Table 3.4: Average AUC scores in percentage (%) for the *seen* setting.

To test the significance of the results, we first perform Friedman test [58]. With a p-value less than 0.001 ($p = 1.96 \times 10^{-4}$), we reject the null hypothesis

and find out a highly significant difference among the methods. We then proceed with Nemenyi test and find the critical distance (CD) as 2.02 for $p = 0.05$. Figure 3.1 reports the results of Nemenyi significance test in terms of average rank of the AUC results reported in Table 3.4. In this figure, connected groups of methods with a line are not significantly different ($p > 0.05$) for the seen setting. Our method is significantly better ($p < 0.05$) than HBOS, OC-SVM and PCC methods, and our method is as competitive as IF and DAE methods for the seen setting.

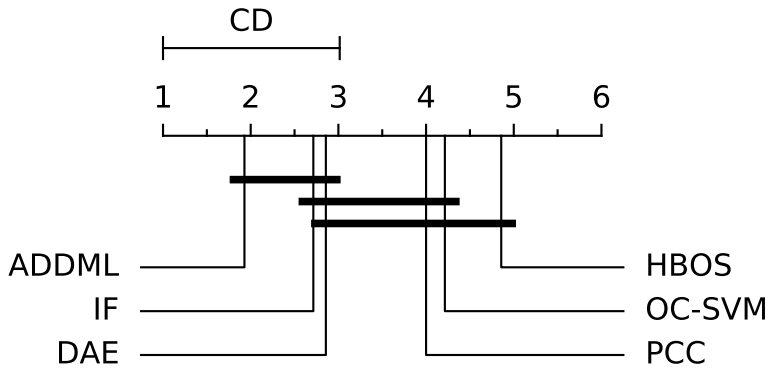

Figure 3.1: Comparison of detectors in terms of average rank with the Nemenyi test for the *seen* setting.

3.5.2 Comparison in the *Unseen* Setting

Table 3.4 shows the results in the seen setting over 9 experiments per model for each dataset. Among six different models on 14 different datasets, our method takes first place on 10 datasets, second place on 3 datasets and sixth place on 1 dataset. On average, our method improves the AUC in absolute between 5.73%-11.47% for the unseen setting.

To test the significance of the results, we first perform Friedman test [58]. With a p-value less than 0.001 ($p = 2.34 \times 10^{-5}$), we reject the null hypothesis and find out a highly significant difference among the methods. We then proceed with Nemenyi test and find the critical distance (CD) as 2.02 for $p = 0.05$.

Dataset	DAE	IF	PCC	OC-SVM	HBOS	ADDML
Gisette	73.76	49.14	73.63	74.24	35.64	**79.21**
Glass	56.91	70.28	44.20	53.13	67.12	**74.78**
Ionosphere	85.45	85.44	79.76	84.29	65.77	**93.29**
Isolet	51.30	54.86	51.36	55.66	54.04	**59.50**
Letter	50.80	62.77	52.08	60.00	55.93	**81.17**
Madelon	50.60	51.14	50.65	50.71	**53.00**	52.46
Magic-Telescope	69.58	**77.85**	68.88	72.55	73.83	74.58
Mnist	85.80	79.81	84.81	84.98	65.64	**86.43**
Musk	**100.00**	99.91	99.98	**100.00**	**100.00**	**100.00**
Pendigits	93.68	**94.80**	92.48	93.13	92.52	86.16
Satellite	61.84	70.58	62.86	66.40	68.34	**80.89**
Satimage-2	97.79	99.45	97.40	99.67	97.81	**99.88**
Shuttle	98.99	**99.67**	99.05	99.17	98.42	99.20
Vowels	56.95	75.44	59.09	77.76	63.34	**84.36**
Average	73.82	76.51	72.59	76.55	70.81	**82.28**

Table 3.5: Average AUC scores in percentage (%) for the *unseen* setting.

Figure 3.2 reports the results of Nemenyi significance test in terms of average rank of the AUC results reported in Table 3.5. In this figure, connected groups of methods with a line are not significantly different ($p > 0.05$) for the unseen setting. Our method is significantly better ($p < 0.05$) than HBOS, OC-SVM and PCC methods, and our method is as competitive as IF and DAE methods for the unseen setting.

3.5.3 Comparison in the *One-Class* Setting

Table 3.4 shows the results in the seen setting over 9 experiments per model for each dataset. Among six different models on 14 different datasets, our method takes first place on 12 datasets and second place on 2 datasets. On average, our method improves the AUC in absolute between 4.44%-11.74% for the one-class setting.

To test the significance of the results, we first perform Friedman test [58]. With a p-value less than 0.001 ($p = 2.31 \times 10^{-7}$), we reject the null hypothesis and find

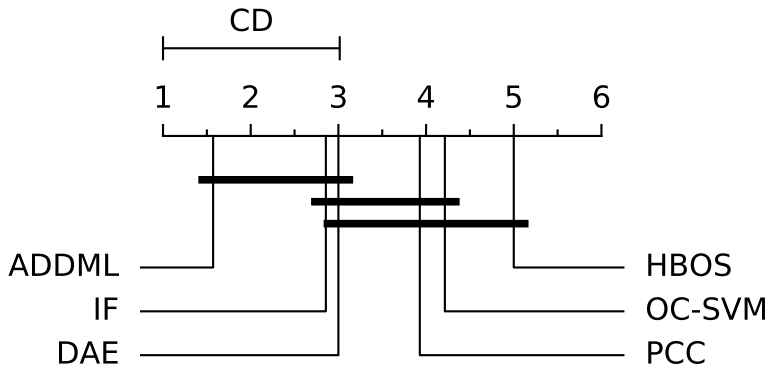

Figure 3.2: Comparison of detectors in terms of average rank with the Nemenyi test for the *unseen* setting.

out a highly significant difference among the methods. We then proceed with Nemenyi test and find the critical distance (CD) as 2.02. Figure 3.3 reports the results of Nemenyi significance test in terms of average rank of the AUC results reported in Table 3.6. In this figure, connected groups of methods with a line are not significantly different ($p > 0.05$) for the one-class setting. Our method is significantly better ($p < 0.05$) than HBOS, DAE and PCC methods, and our method is as competitive as IF and OC-SVM methods for the one-class setting.

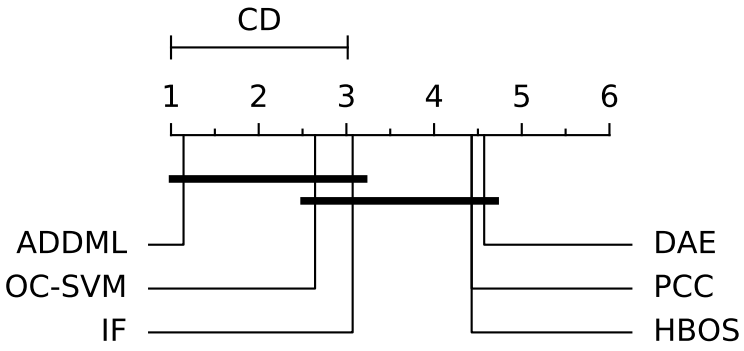

Figure 3.3: Comparison of detectors in terms of average rank with the Nemenyi test for the *one-class* setting.

Dataset	DAE	IF	PCC	OC-SVM	HBOS	ADDML
Gisette	81.94	48.96	81.86	82.06	37.48	**84.22**
Glass	59.13	**74.89**	47.14	62.57	71.68	71.73
Ionosphere	90.96	91.01	89.55	**95.93**	75.00	95.81
Isolet	52.10	55.55	52.11	56.94	54.61	**61.16**
Letter	51.66	62.21	52.66	61.56	60.08	**81.50**
Madelon	51.01	51.99	51.03	51.10	52.85	**53.77**
Magic-Telescope	70.36	77.53	74.83	73.93	73.26	**80.80**
Mnist	90.86	86.36	90.35	90.80	70.33	**91.25**
Musk	**100.00**	95.88	**100.00**	**100.00**	**100.00**	**100.00**
Pendigits	94.24	96.71	94.12	96.46	93.59	**97.93**
Satellite	67.73	79.47	69.57	75.71	77.85	**81.92**
Satimage-2	97.93	99.31	97.58	99.70	97.25	**99.84**
Shuttle	99.36	99.61	99.44	99.62	98.71	**99.96**
Vowels	59.22	76.80	61.20	82.30	63.82	**90.90**
Average	76.18	78.31	75.82	80.62	73.32	**85.06**

Table 3.6: Average AUC scores in percentage (%) for the *one-class* setting.

3.5.4 Overall Comparison

Among six methods, the ADDML achieves the best performance for 9 times in the seen setting, 10 times in the unseen setting, and for 12 times in the one-class setting out of the 14 datasets. On average, the ADDML improves the AUC in absolute between 4.71%-10.59% for the seen setting, 5.73%-11.47% for the unseen setting, and 4.44%-11.74% for the one-class setting. Thus, our method performs significantly better than the compared well-known methods in the literature.

The ADDML achieves the highest scores in all settings for all datasets with dimensions higher than or equal to 100, i.e., *Mnist*, *Musk*, *Gisette* (except the seen setting), *Madelon* (except the seen and the unseen settings), and *Isolet*. In the seen and the unseen settings, which contains outliers in the training data, the ADDML achieves the highest score for the datasets higher than 10% outlier rate, i.e., the *Ionosphere*, and the *Satellite*. These results demonstrate that our method achieves outstanding results compared to the well-known methods in the literature when the number of dimensions is high, or when the outlier rate is high.

Similar to the other methods, the ADDML achieves a higher or equal score in the one-class setting compared to its seen and *Unseen* settings for 12 out of the 14 datasets. Furthermore, when the one-class score is lower than the seen setting or the unseen setting, the score of the one-class setting is very close to the other settings. Thus, all methods better model the data as the number of anomalies decrease. In the following section, we analyze the data distillation hyperparameter.

For all settings, our method's average AUC scores are higher than other methods with at least 4.44% (absolute) AUC. In terms of statistical significance, our method is significantly better than HBOS and PCC and as competitive as IF for all settings. Our method is significantly better than DAE for the one-class setting and as competive as DAE for other settings. Our method is as competitive as OC-SVM for the one-class setting and significantly better than OC-SVM for other settings.

3.6 Analysis of the Data Distillation Hyperparameter ρ^n

In this section, we analyze the normal ratio parameter ρ^n for the data distillation, which defines the ratio of the enclosed instances in the normal enclosure region. Recall that only ρ^n of the instances, which are likely to be normal, are used at each epoch to train our method. We use the instance closeness loss with $\rho^h = \frac{1}{3}$ as in Section 3.5 for all experiments. We use the evaluation and the cross-validation method described in Section 3.1.

Fig. 3.4 illustrates the AUC for the *Letter* dataset with respect to the different ρ^n values. In the one-class setting, we do not achieve a significant improvement by changing ρ^n. Hence, it is stable, i.e., at most $\approx 2\%$ absolute change in AUC, for $\rho^n > 0.5$ in the one-class setting showing that our method does not require hand tuning and robust. Recall that our motivation for data distillation is to clean

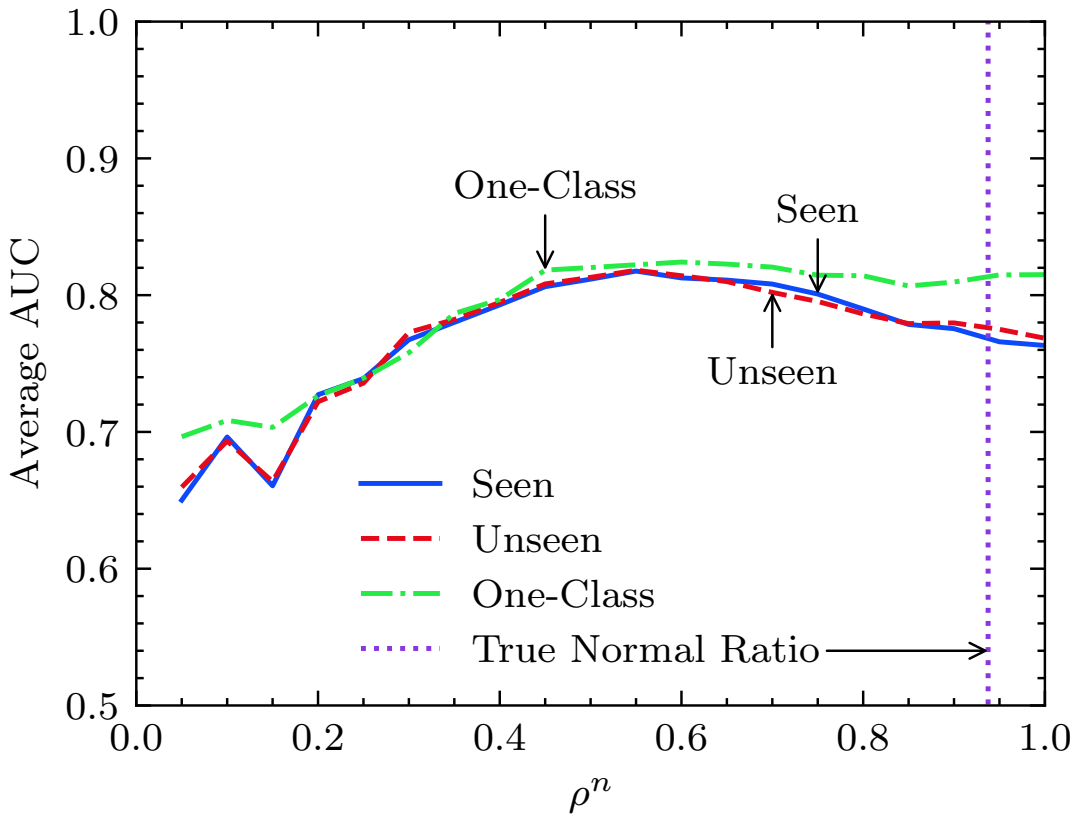

Figure 3.4: The AUC scores for the *Letter* dataset with respect to the normal ratio $\rho^n \in [0.05, 1.00]$ with spacing 0.05 for ρ^n on the x-axis.

the training data from anomalies. Thus, the stableness with respect to the ρ^n is advantageous when one does not know whether training data contains anomalies. Our method provides lesser performance for seen and unseen settings when there is no or less data distillation, i.e., $\rho^n \approx 1.0$. As ρ^n decreases, the performance increases since the method use the instances that are more likely to be normal. Thus, the model becomes more robust to the outliers. The performance reaches its peak at near 0.5 instead of the true normal rate, which is 0.9375. Naturally, since our method is unsupervided, it distills the data based on self-supervision instead of the true labels, which is expected. The scores of the ADDML for all ρ^n values in Fig. 3.4 are higher than the scores of the other methods defined in Section 3.5 for the *Letter* dataset.

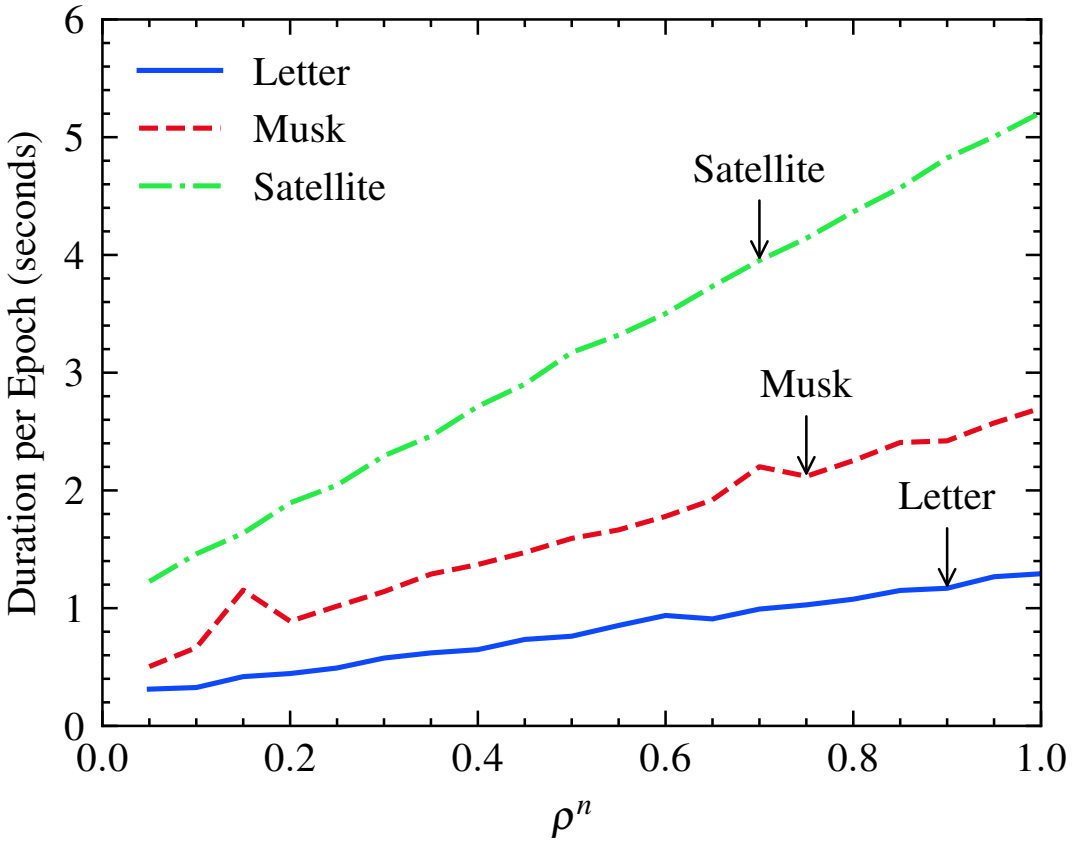

Figure 3.5: Average durations per epoch for varying data distillation ratio ρ^n.

Figure 3.5 shows average duration per epoch including validation for varying data distillation ratios. Figure 3.5, shows linear regimes when varying ρ^n since

we change the number of instances to be used for backpropagation by changing ρ^n. Average epoch duration is shortened by decreasing ρ^n and lengthened by increasing ρ^n.

In the following section, we analyze the hard normal mining hyperparameter.

3.7 Analysis of the Hard Normal Mining Hyperparameter ρ^h

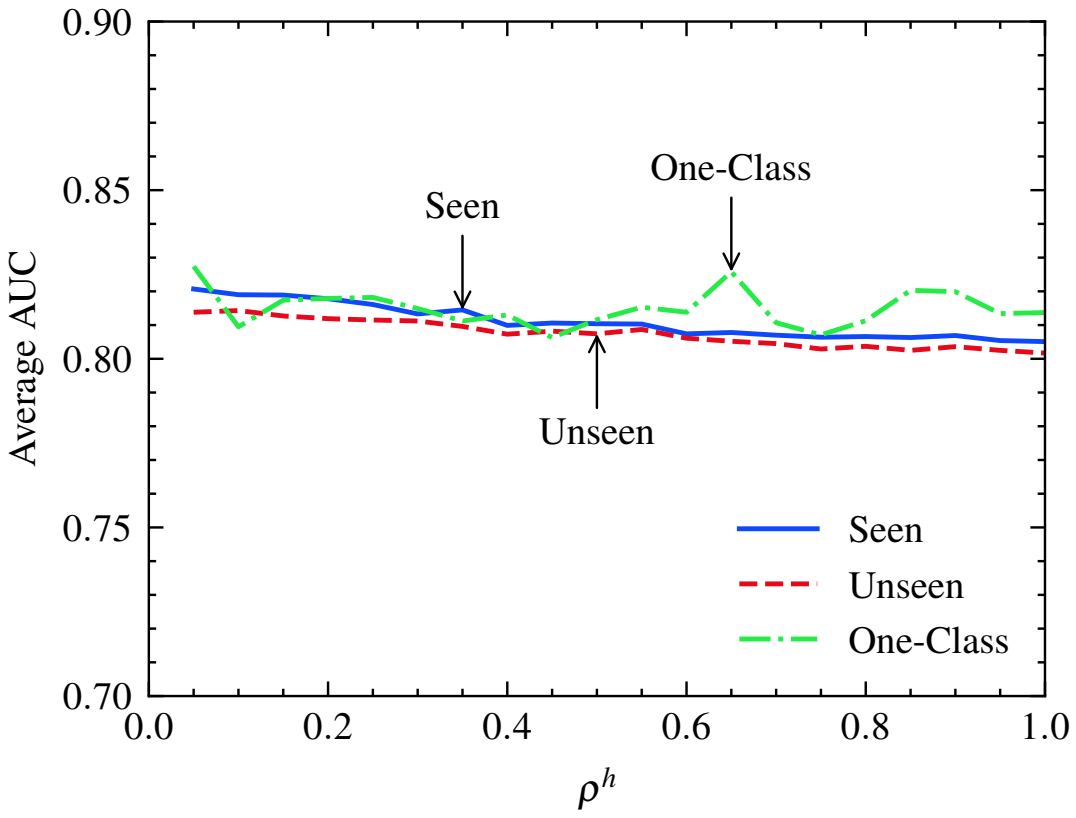

Figure 3.6: The AUC scores for the *Letter* dataset with respect to hard mining ratio $\rho^h \in [0.05, 1.00]$ with spacing 0.05 for ρ^h on the x-axis.

In this section, we analyze the hard normal mining hyperparameter ρ^h, which defines the ratio of the largest distances to be included in the loss for the weight updates. As in Section 3.5, we use the instance closeness loss with $\rho^n = \frac{2}{3}$ for

the seen and the unseen settings and $\rho^n = 1$ for the one-class setting for all experiments. We use the evaluation and the cross-validation method described in Section 3.1.

Fig. 3.6 illustrates the AUC for the *Letter* dataset with respect to the different ρ^h values. The method seems robust with respect to the change in ρ^h hyperparameter for all settings. The method achieves the highest scores for $\rho^h < 0.35$ with slight differences. Reducing ρ^h increases the AUC slightly for the seen and the unseen settings. For the one-class setting, the method fluctuates for some values of ρ^h, although not significant. Thus, using $\rho^h < 0.35$ seems advantageous due to the running time gain for all settings and a slight performance gain for the seen and the unseen settings. The scores of the ADDML for all ρ^h values in Fig. 3.6 are higher than the scores of the other methods defined in Section 3.5 for the *Letter* dataset.

3.8 Analysis of the Model Collapse

In this section, we empirically analyze the behavior of our model considering the model collapse problem by analyzing losses and the sum of absolute weights.

Recall that our epoch limit is $n = 50$, and we stop training when this limit is reached. Suppose validation loss stops improving for $e^v = 5$ epochs. In that case, we stop training before the epoch limit and use the model with the minimum validation loss. To create the figures in this section, we use only the first run of nine runs in total. We also use the hyperparameters described in Section 3.2.

Figure 3.7 shows the average training and validation losses throughout epochs on the *Letter* dataset. Our model's average validation loss and average training loss decrease throughout the training. Thus, our model stops when the epoch limit ($n = 5$) is reached. When we increase this limit to $n = 1000$, the model stops training at the 336th epoch. Moreover, when $n = 1000$, our method achieves 80.45% AUC in the seen setting, 81.10% AUC in the unseen setting, and 81.15%

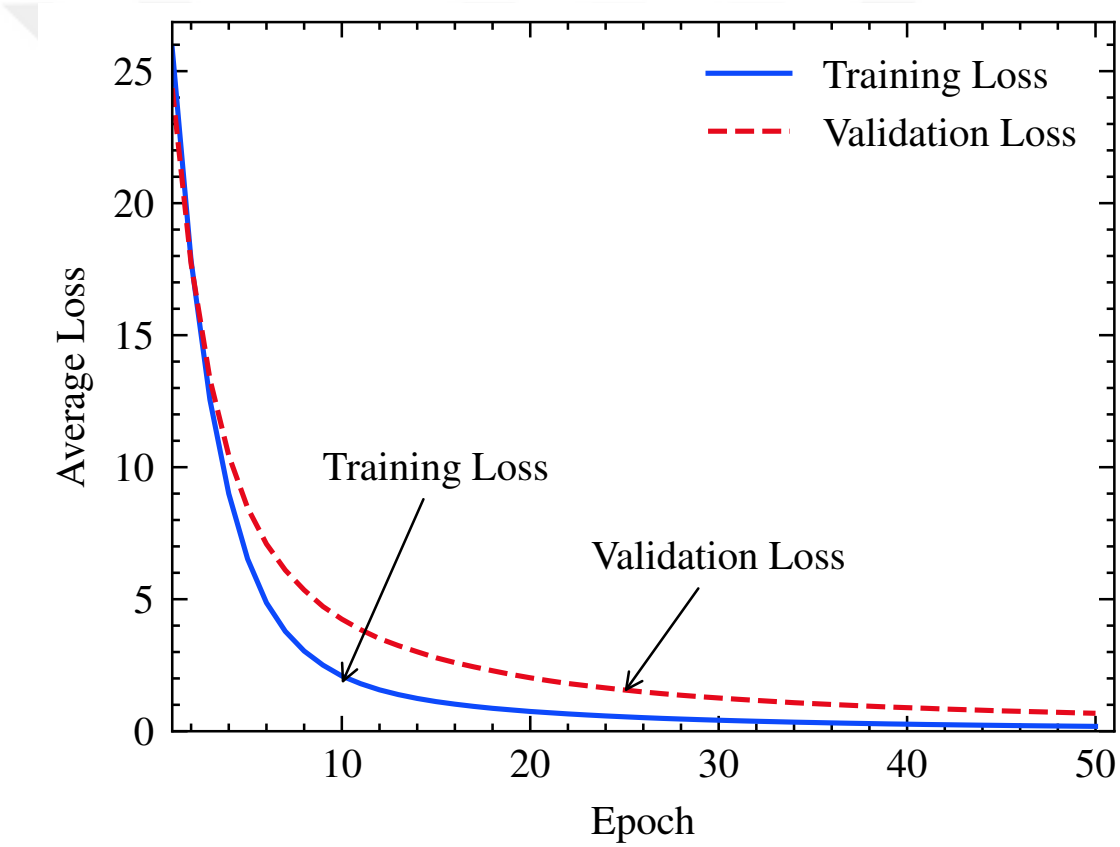

Figure 3.7: Analysis of model collapse for the *Letter* dataset

AUC in the one-class setting. Increasing epoch limit to 1000 from 50 reduces AUC in absolute by only 1.04% in the seen setting, 0.02% in the unseen setting, and 0.35% in the one-class setting. These insignificant differences between different epoch limits indicate the effectiveness of early stopping.

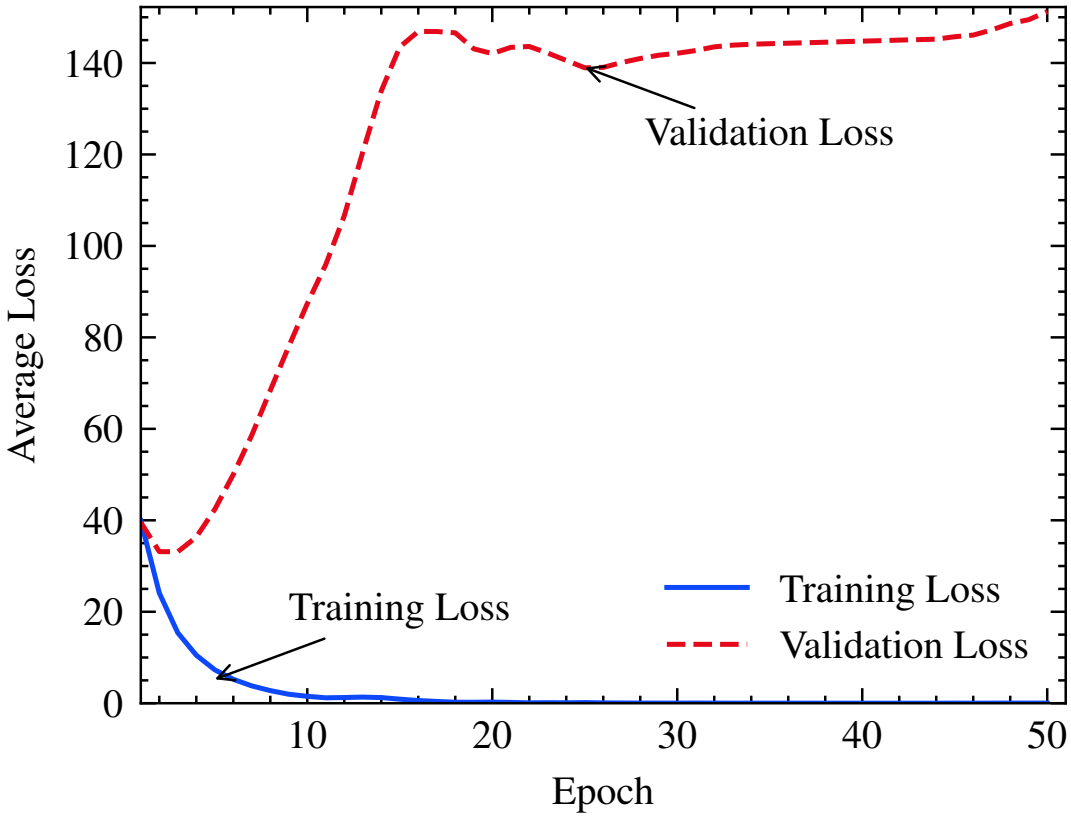

Figure 3.8: Analysis of model collapse for the *Gisette* dataset

Figure 3.8 shows the average training and validation losses throughout epochs on the *Gisette* dataset. For this dataset, early stopping occurs at the end of the eighth epoch, and thus the model from the third epoch is used since it has the lowest validation loss. After the 3rd epoch, the average validation loss grows, whereas the average training loss approaches zero.

Figure 3.9, presents the average of absolute weights throughout epochs. Although weights are typically expected to collapse to zero due to the model collapse problem, the average of absolute model weights increases throughout the epochs. According to this figure, the model does not collapse to all-zero-weights.

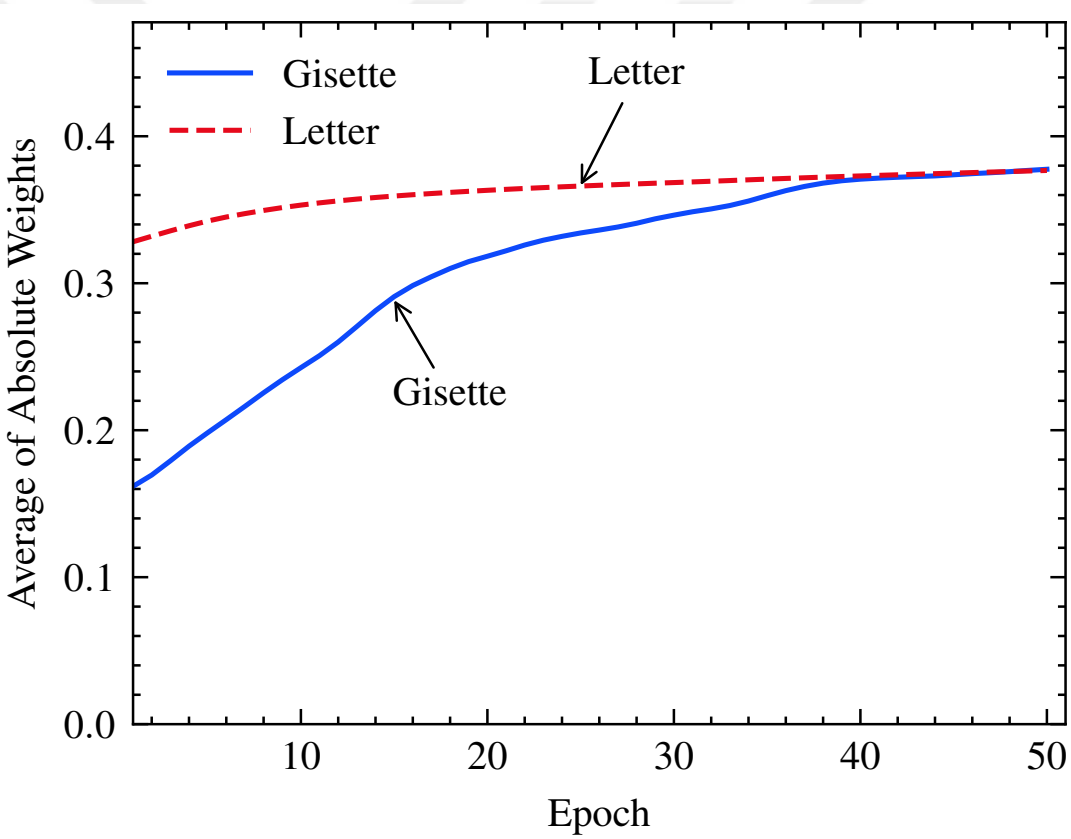

Figure 3.9: Analysis of model weights

Additionally, as seen in Section 3.7, the online hard normal mining component improves the model as ρ^h gets lower, which may be due to its support to the model collapse problem. Online hard normal mining does not incorporate instances to the training that are very close to their center. Thus, this component also helps to reduce the effect of model collapse.

Chapter 4

Conclusion

We have studied anomaly detection on multivariate data. In Chapter 2, we have introduced a novel anomaly detection method using the DML. Our method projects the data into a possibly lower-dimensional latent space by minimizing the similarity loss, which aims to bring similar instances closer in the metric space, where the model is optimized end-to-end. For the test stage, we derive a scoring function with $\Theta(1)$ time and memory complexity, i.e., the distance to the center, to score the outlierness of the instances. Our approach is generic so that it can be adapted to any type of data such as images or time series by only changing the underlying neural network architecture, for which we provide such adaptations as remarks. Moreover, our method is generic so that one can readily extend our approach using different ideas in the DML literature. In Chapter 3, we have demonstrated that our method achieves significant performance improvements compared to the state-of-the-art methods through an extensive set of experiments in various real-world datasets. We have also performed ablation study and hyperparameter studies. As future work, more recent deep metric learning losses can be adapted to the unsupervised anomaly detection through the methodology introduced in this thesis. Moreover, the model collapse problem discussed in Section 3.8 can be further analyzed.

Bibliography

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[2] P. Wu, J. Liu, and F. Shen, "A deep one-class neural network for anomalous event detection in complex scenes," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[3] Y. Cao, Y. Li, S. Coleman, A. Belatreche, and T. M. McGinnity, "Adaptive hidden markov model with anomaly states for price manipulation detection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 2, pp. 318–330, 2014.

[4] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[5] H. Ozkan, O. S. Pelvan, and S. S. Kozat, "Data imputation through the identification of local anomalies," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 10, pp. 2381–2395, 2015.

[6] C. C. Aggarwal, "Outlier analysis," in *Data Mining*, pp. 237–263, Springer, 2015.

[7] B. Harwood, B. Kumar, G. Carneiro, I. Reid, T. Drummond, *et al.*, "Smart mining for deep metric learning," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2821–2829, 2017.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[9] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[10] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[12] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[13] P. Druzhkov and V. Kustikova, "A survey of deep learning methods and software tools for image classification and object detection," *Pattern Recognition and Image Analysis*, vol. 26, no. 1, pp. 9–15, 2016.

[14] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *ieee Computational intelligenCe magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[15] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European Conference on Computer Vision*, pp. 850–865, Springer, 2016.

[16] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1735–1742, IEEE, 2006.

[17] X. Yang, P. Zhou, and M. Wang, "Person reidentification via structural deep metric learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 10, pp. 2987–2998, 2019.

[18] C. Li, C. Liu, L. Duan, P. Gao, and K. Zheng, "Reconstruction regularized deep metric learning for multi-label image classification," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2019.

[19] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3109–3118, 2015.

[20] M. Faraki, M. T. Harandi, and F. Porikli, "Large-scale metric learning: A voyage from shallow to deep," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 9, pp. 4339–4346, 2017.

[21] Y. Suh, B. Han, W. Kim, and K. M. Lee, "Stochastic class-based hard example mining for deep metric learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7251–7259, 2019.

[22] M. Masana, I. Ruiz, J. Serrat, J. van de Weijer, and A. M. Lopez, "Metric learning for novelty and anomaly detection," *arXiv preprint arXiv:1808.05492*, 2018.

[23] B. Du and L. Zhang, "A discriminative metric learning based anomaly detection method," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 11, pp. 6844–6857, 2014.

[24] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[25] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[26] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire, "Novelty detection using level set methods," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 576–588, 2015.

[27] L. Livi, A. Sadeghian, and W. Pedrycz, "Entropic one-class classifiers," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 12, pp. 3187–3200, 2015.

[28] M. Moshtaghi, T. C. Havens, J. C. Bezdek, L. Park, C. Leckie, S. Rajasegarar, J. M. Keller, and M. Palaniswami, "Clustering ellipses for anomaly detection," *Pattern Recognition*, vol. 44, no. 1, pp. 55–69, 2011.

[29] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 427–438, 2000.

[30] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.

[31] M. Sabokrou, M. Fathy, G. Zhao, and E. Adeli, "Deep end-to-end one-class classifier," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2020.

[32] T. Ergen and S. S. Kozat, "Unsupervised anomaly detection with lstm neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[33] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International conference on machine learning*, pp. 4393–4402, 2018.

[34] D. Pérez-Cabo, D. Jiménez-Cabello, A. Costa-Pazo, and R. J. López-Sastre, "Deep anomaly detection for generalized face anti-spoofing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[35] I. Chingovska, A. Anjos, and S. Marcel, "On the effectiveness of local binary patterns in face anti-spoofing," in *2012 BIOSIG-proceedings of the international conference of biometrics special interest group (BIOSIG)*, pp. 1–7, IEEE, 2012.

[36] R. C. Aygun and A. G. Yavuz, "Network anomaly detection with stochastically improved autoencoder based models," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 193–198, IEEE, 2017.

[37] I. Delibalta, K. Gokcesu, M. Simsek, L. Baruh, and S. S. Kozat, "Online anomaly detection with nested trees," *IEEE Signal Processing Letters*, vol. 23, no. 12, pp. 1867–1871, 2016.

[38] K. Gokcesu and S. S. Kozat, "Online anomaly detection with minimax optimal density estimation in nonstationary environments," *IEEE Transactions on Signal Processing*, vol. 66, no. 5, pp. 1213–1227, 2017.

[39] M. U. Ndubuaku, A. Anjum, and A. Liotta, "Unsupervised anomaly thresholding from reconstruction errors," in *International Conference on Internet and Distributed Computing Systems*, pp. 123–129, Springer, 2019.

[40] A. Akhriev and J. Marecek, "Deep autoencoders with value-at-risk thresholding for unsupervised anomaly detection," in *2019 IEEE International Symposium on Multimedia (ISM)*, pp. 208–2083, IEEE, 2019.

[41] M.-H. Chang, C. Chen, D. Das, and M. Pecht, "Anomaly detection of light-emitting diodes using the similarity-based metric test," *IEEE Transactions on industrial informatics*, vol. 10, no. 3, pp. 1852–1863, 2014.

[42] B. Kulis *et al.*, "Metric learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 5, no. 4, pp. 287–364, 2013.

[43] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[45] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[46] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[47] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[48] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire, "An experimental evaluation of novelty detection methods," *Neurocomputing*, vol. 135, pp. 313–327, 2014.

[49] D. L. Streiner and J. Cairney, "What's under the roc? an introduction to receiver operating characteristics curves," *The Canadian Journal of Psychiatry*, vol. 52, no. 2, pp. 121–128, 2007.

[50] C. D. Brown and H. T. Davis, "Receiver operating characteristics curves and related decision measures: A tutorial," *Chemometrics and Intelligent Laboratory Systems*, vol. 80, no. 1, pp. 24–38, 2006.

[51] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, IEEE, 2008.

[52] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "Principal component-based anomaly detection scheme," in *Foundations and Novel Approaches in Data Mining*, pp. 311–329, Springer, 2006.

[53] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm," *KI-2012: Poster and Demo Track*, pp. 59–63, 2012.

[54] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[55] E. Manzoor, H. Lamba, and L. Akoglu, "xstream: Outlier detection in feature-evolving data streams," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1963–1972, 2018.

[56] S. Rayana, "{ODDS} library," 2016.

[57] T. Pevnỳ, "Loda: Lightweight on-line detector of anomalies," *Machine Learning*, vol. 102, no. 2, pp. 275–304, 2016.

[58] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.