Deep Unsupervised Anomaly Detection

Tangqing Li\textsuperscript{1}, Zheng Wang\textsuperscript{2}, Siying Liu\textsuperscript{2}, and Wen-Yan Lin\textsuperscript{3}

\textsuperscript{1}National University of Singapore, \textsuperscript{2}Institute for Infocomm Research, Singapore, \textsuperscript{3}Singapore Management University

li_tangqing@u.nus.edu, \{zhwang, liusy1\}@i2r.a-star.edu.sg, daniellin@smu.edu.sg

Abstract

This paper proposes a novel method to detect anomalies in large datasets under a fully unsupervised setting. The key idea behind our algorithm is to learn the representation underlying normal data. To this end, we leverage the latest clustering technique suitable for handling high dimensional data. This hypothesis provides a reliable starting point for normal data selection. We train an autoencoder from the normal data subset, and iterate between hypothesizing normal candidate subset based on clustering and representation learning. The reconstruction error from the learned autoencoder serves as a scoring function to assess the normality of the data. Experimental results on several public benchmark datasets show that the proposed method outperforms state-of-the-art unsupervised techniques and is comparable to semi-supervised techniques in most cases.

1. Introduction

Anomaly detection refers to the identification of patterns that do not conform to expected normal behavior [6]. It is a critical task in diverse application domains such as fraud detection [23], intrusion detection [16] and surveillance video profiling [31, 25]. While the concept of an anomaly is intuitively easy for humans to understand, it is hard to define mathematically. Fundamentally, an anomaly is something with insufficient similarity to the rest of the data. This similarity can be computed on the basis of some feature difference. However, what makes an ideal feature representation for the data depends on what constitutes an anomaly. This forces anomaly detection into a chicken-or-egg problem in which there are a pair of problems, neither of which can be solved before the other.

To date, a number of works have attempted this problem by training an autoencoder to create low-dimensional representations for anomaly detection [5, 33, 35]. The anomalies are rejected and the autoencoder retrained [22, 29]. While this gives reasonable results, it is fundamentally dependent on how well the first iteration solves the problem.

We propose a solution in which anomalies can be defined using approximately correct features. This is achieved through an observation. Given a feature, anomalies approximately correspond to instances of high variance distributions. Such instances can be identified using a distribution-clustering [19] framework. This hypothesis provides a reliable starting point for normal data selection. We train an autoencoder from the normal data subset obtained from distribution-clustering, and iterate between hypothesizing normal candidate subset and representation learning. The reconstruction error serves as a scoring function to assess the normality of the data. The proposed framework does not rely on any training labels. Instead, it iteratively distills out anomalous data and improves the learned representation of normal data by incorporating clustering techniques into the process. Our method works with the least assumption on the data itself and does not use any label information, even in the training phase. The only assumption is that the anomalies are not statistically dominant in the entire dataset; and for this exact reason they are anomalies by nature.

We extensively assess the broad applicability of the proposed model on network intrusion, image and video data. Empirical results show that the proposed method outperforms the existing state-of-art approaches in terms of both accuracy and robustness to the percentage of anomalous data.

2. Related Works

Existing anomaly detection methods can be grouped into three categories.

Reconstruction-based method These methods assume that anomalies are in-compressible and thus cannot be effectively reconstructed from low-dimensional projections. Classical methods like Principle Component Analysis(PCA) [13] and Robust-PCA [4] are motivated by this assumption. In recent works, different forms of deep autoencoder are
proposed to analyze the reconstruction error. Xia et al. [30] show that by introducing a regularizing term to a convolutional autoencoder, the anomalies tend to produce a bigger reconstruction error. Variational Autoencoder (VAE) [1] and Generative Adversarial Networks (GANs) [26] have also been introduced to perform reconstruction-based anomaly detection. These methods demonstrate promising results when the anomaly ratio is fairly low. Although the reconstruction of anomalous samples, based on a reconstruction scheme optimized for normal data, tends to generate a higher error, a significant amount of anomalous samples could mislead the autoencoders to learn the correlations in the anomalous data instead. Padhorskiy et al. have [24] adopted Adversarial Autoencoder [20] for generative probabilistic novelty detection. These methods, although claimed as unsupervised, require pre-isolation/identification of classes of normal data in the training phase, as normal data is needed to describe the inlier distribution. For example, In GAN methods [26, 24], label information is used to feed normal data into the discriminator during training.

Density estimation, representation learning and clustering Motivated by the assumption that anomalies occur less frequently, these algorithms treat anomalies as low-density regions in some feature space. Clustering analysis, such as Robust-KDE [14], is often used for density estimation and anomaly detection. Unfortunately, due to the curse of dimensionality, these methods are less applicable to analyzing high-dimensional data, where density estimation is a challenge in itself.

A two-step approach is normally adopted to counter this issue, where dimensionality reduction is conducted first, followed by clustering analysis as a separate step. One drawback of this approach is that dimensionality reduction is trained without the guidance from the subsequent clustering analysis; hence the key information for clustering analysis could be lost during dimensionality reduction. Recently, Ionescu et al. [12] proposed to train autoencoders on tracked objects in videos to detect anomalous events. The latent representations from autoencoders are clustered, followed by a one-versus-rest classifier to discriminate between the formed clusters. There are also works that jointly learn dimensionality reduction and clustering components based on deep autoencoder [33, 35]. Notably, DAGMM [35] utilizes an autoencoder to generate a low-dimensional representation and its reconstruction error, which is further fed into an estimation network based on Gaussian Mixture Model (GMM). However, as its autoencoder was trained on the whole dataset, it is vulnerable to a high percentage of anomalous samples and may learn wrong correlations. In contrast, our proposed method addresses this issue by first finding a normal candidate subset to train an autoencoder and then iterating between representation learning and refinement of the normal candidate.

One-class classification One-class SVM [9, 5] is widely used. Under this framework, a discriminative boundary surrounding the normal instances is learned by algorithms. However, when dimensionality goes higher, such techniques often suffer from suboptimal performance due to the curse of dimensionality. OCNN [5] attempts to circumvent this problem by using an autoencoder for dimensionality reduction. However, OCNN requires training data with relatively low anomaly ratio, in order to obtain an optimized NN model to differentiate anomalies from single-class normal data. Zenati et al. [32] use GAN to learn a generative model from the normal data, and leverage the latent representation of the generator input or from the encoder in the discriminator learning. Label information of the normal data is required for training.

3. Problem Formulation

Let \( X = \{x_i\}, i = 1, \ldots, N, x \in \mathbb{R}^k \) be the set of input data points that contains a certain percentage of anomaly. The goal of anomaly detection is to learn a scoring function \( h(x), h : \mathbb{R}^k \mapsto \mathbb{R} \), to classify samples \( x_i \) based on some threshold \( \lambda \):

\[
y_i = \begin{cases} 0, & \text{if } h(x_i) < \lambda \\ 1, & \text{if } h(x_i) \geq \lambda \end{cases} \tag{1}
\]

where \( y_i \) are the labels, \( y_i = 0 \) indicates \( x_i \) is normal and \( y_i = 1 \) indicates anomalous.

An overview of the proposed end-to-end anomaly detection system is presented in Fig. 1. The major component of this system is an autoencoder that learns a low-dimensional representation of the input data that are often of high dimensions, to enable simplified modeling of the underlying distribution of the data. Under a fully unsupervised setting, the only information we are given is the set of input data \( X \), without any label information. As an initialization, we leverage the latest clustering technique for high-dimensional data [19] to provide soft supervisory signals.

Since our input data is unlabelled, we derive a “training” set \( S_{\text{train}} \), where \( S_{\text{train}} \subset X \) based on the following:

\[
S_{\text{train}} = C(X, p_0) \tag{2}
\]

where \( C \) represents a selection process based on clustering output, and \( p_0 \) represents the percentage of anomaly, it controls which are the clusters to be accepted into the “training” set. In our experiments, we compute the threshold as the \( (100 - p_0)^{th} \) percentile of cluster variance, and accept clusters with variance smaller than this threshold. The assumption here is that clusters with large variance are likely to contain anomalous members.
we adopt a more conservative cluster variance threshold.

work on the entire input set $X$ where $\Theta$ set the input with its decoded counter-part. Using the training input data; and (2) the reconstruction error by comparing input data points more distinguishable.

3.1. Scoring Function Learning

The autoencoder network provides two sources of features: (1) a low-dimensional representation of the original input data; and (2) the reconstruction error by comparing the input with its decoded counter-part. Using the training set $S_{train} = \{s_1,s_2,\cdots,s_M\}$, the autoencoder learns the encoding function $f_{en}$:

$$z_c = f_{en}(s; \Theta_{en}), \quad \forall s \in S_{train}, \quad z_c \in \mathbb{R}^{k_{bn}}$$

where $\Theta_{en}$ are the learned parameters for the encoder. $z_c$ are known as the bottle-neck features of dimension $k_{bn}$.

Similarly, for the decoding part, we have:

$$x' = f_{de}(z_c; \Theta_{de}),$$

where $\Theta_{de}$ are the learned parameters for decoding. $x'$ are the reconstructed features.

Upon training, we have a learned autoencoder with optimized parameters $\{\Theta_{en}, \Theta_{de}\}$. We apply the encoder network on the entire input set $X$ to produce a new set of features $Z = \{z_1,z_2,\cdots,z_N\}$. Each data point in this set is formed by concatenating the bottle-neck feature with the reconstruction error:

$$z = [z_c; z_r], \quad z \in \mathbb{R}^{k_{bn}+1},$$

where the reconstruction error $z_r$ is measured in terms of cosine similarity between $x$ and its decoded counter-part:

$$z_r = d(x, x') = \cos^{-1}\left(\frac{x^T x'}{\|x\| \|x'\|}\right)$$

$\ell_2$ normalization is applied on each data point $z$. The inclusion of reconstruction loss helps to make anomalous data points more distinguishable. $Z$ is now of a much lower dimension than the input data $X$. Hence, traditional clustering techniques such as Gaussian Mixture Model would suffice for subsequent training set selection. To ensure the initial training set can capture most of the normal samples, we adopt a more conservative cluster variance threshold.

With the new encoding scheme, the entire input set $X$ is now represented as $Z$. We can "re-label" the training set $X$ by using $Z$ as a proxy, and an assumed anomaly percentage $p$ to determine the threshold. Similar to the initial training set selection, we select members that belong to low-variance clusters in $Z$. The process of $Training set selection \rightarrow Autoencoder training \rightarrow New feature computation$ is performed iteratively. The training set is updated as follows:

$$Z_{train}^{t+1} = C(Z^t, p),$$

$$S_{train}^{t+1} = \{x_j : \forall z_j \in Z_{train}^{t+1}\},$$

where the superscript $t$ here refers to the $t^{th}$ iteration.

Finally, the training process terminates when there is no change in the set of selected normal samples between two successive iterations. After the last iteration, $t = t_F$, we obtain the autoencoder parameters $\{\Theta^{t_F}_{en}, \Theta^{t_F}_{de}\}$, and use it to construct the scoring function:

$$h(x) = d(x, x') = d(x, f_{de}(f_{en}(x; \Theta^{t_F}_{en}); \Theta^{t_F}_{de})), $$

where $x'$ is the result of going through the encoding-decoding process according to the trained autoencoder.

3.2. Algorithm

The proposed framework is summarized in Algorithm 1. We obtain an initial split of the data into normal and
abnormal subsets through clustering (i.e. GMM for KDD-CUP data and Distribution Clustering [19] for image and video data). Candidate normal samples are then passed into the autoencoder for representation learning. All examples’ memberships are re-evaluated based on its low-dimensional representation every $r$ epochs, where the new normal candidates are fed into the autoencoder for learning. Finally, when there is no change in all samples’ memberships, an encoder that learns the low-dimensional projection of the normal data is finalized, and its reconstruction loss will be used for scoring.

**Initialization**  We use Distribution Clustering [19] to make an educated guess about the normal data subset for image and video data. Selected clustering outputs for CIFAR-10 and MNIST datasets are shown in Fig. 2. Observe that as cluster variance increases, the samples’ appearance become more anomalous.

**Convergence**  Assuming a $p\%$ anomaly percentage, our algorithm starts with a tight cut-off, accepting clusters with variances below $(100 - p_0)^{th}$ percentile as an initial training set, where $p_0 > p$. This ensures the initial training set is as pure as possible. Our assumption is that given partial normal data, the autoencoder would be able to learn a representation and generalize well on the “unseen” normal data that was discarded, and progressively recover them as iterations go on. Empirically, we plotted the AUROC, AUPRC and F-score for 20 iterations for the KDDCUP experiment, presented in Fig. 3. It demonstrates the convergence as iteration progresses. The same behavior was observed throughout our experiments on other datasets.

### 4. Experiments

#### 4.1. Baseline Methods

On the topic of anomaly detection, there are different terminologies concerning the nature of supervision: (a) Algorithm uses label information of the normal class for training (label information could be used in part, or all of the stages of an algorithm); (b) No training labels are given, algorithm treats the entire dataset with both normal and anomalous classes as input. For the purpose of this paper, we term type (a) semi-supervised and type (b) unsupervised. We evaluate our method against the following state-of-the-art methods:

**OC-NN**(semi-supervised) One-class neural networks (OC-NN) [5] contains 2 major components: a deep autoencoder and a feed-forward convolutional network. The deep encoder is trained on normal data for representation learning. The trained encoder, with its parameters frozen, is subsequently used as the input layers of a feed-forward network with 1 extra hidden layer. Variants of OC-NN employ different activation functions (i.e. linear, sigmoid, relu) in the hidden layer. We report the best score attained among all possible activation functions in our experiments.

**OC-SVM**(unsupervised) One-class support vector machine (OC-SVM) [9] is a kernel-based method for anomaly detection. The algorithm searches for best-performing hyperparameters $\gamma$ (kernel coefficient) and $\nu$ (upper bound of the fraction of training errors and lower bound of the fraction of support vectors) to obtain the optimal AUROC [3].

**DAGMM**(unsupervised) Deep autoencoding Gaussian mixture model (DAGMM) [35], comprised of one compression net and one estimation net, is a method based on representation learning. The compression network provides

---

**Algorithm 1 Deep end-to-end Unsupervised Anomaly Detection**

**Input:** $X = \{x_i\}, i = 1, 2, \ldots, N$: set of normal and anomalous input examples. $r$: number of epochs required for re-evaluation of the membership of the entire input set $X$. $p_0$ and $p$: thresholds for initial and subsequent training set selection, respectively

**Output:** Reconstruction-based anomaly score function $h(x)$ and trained autoencoder $\{\Theta_{en}, \Theta_{de}\}$,

1. **procedure** GET_DECISION_SCORE($X, r, p, f_{en}, f_{de}$)
2. $S_{train} \leftarrow C(X, p_0)$  // Run clustering, select instances from low-variance clusters
3. $L = \{k: \forall x_k \in S_{train}\}$  // L is the set of indices of selected normal training samples
4. $L^{old} := \emptyset$
5. **while** setdiff($L^{old}, L$) $\neq \emptyset$ **do**
6. **for** each epoch **do**
7. if $((CurrentEpoch + 1) \mod r) = = 0$
8. $\triangleright$ Re-evaluate normality every $r$ epochs
9. $Z_\epsilon \leftarrow f_{en}(X, \Theta_{en})$  // Bottle-neck features
10. $X' \leftarrow f_{de}(Z_\epsilon, \Theta_{de})$
11. $Z_r \leftarrow d(X, X')$  // Reconstruction error
12. $Z \leftarrow [Z_\epsilon, Z_r]$
13. $S_{train} \leftarrow C(Z, p)$  // Get new training set according to threshold $p$
14. $L^{old} := L$
15. $L \leftarrow S_{train}$  // Update set of indices for training samples
16. **else**
17. $\triangleright$ Train $f_{en}, f_{de}$ on $S_{train}$ to obtain $\{\Theta_{en}, \Theta_{de}\}$
18. **end for**
19. $\Theta_{en}^{p} = \Theta_{en}$, $\Theta_{de}^{p} = \Theta_{de}$
20. **Output** $h(x)$ according to finalized autoencoder $\{\Theta_{en}^{p}, \Theta_{de}^{p}\}$ base on Eq. (9)
21. **end procedure**

---

3639
low-dimensional representations of input samples and the reconstruction error features. They are fed into the estimation network, which functions as a Gaussian Mixture Model, to predict the mixture membership for each sample. We modify the original DAGMM algorithm by adding a small value to the diagonal elements of the covariance matrix. The model achieves better results than the reported score from the original work.

**Deep anomaly detection using geometric transformations (semi-supervised)** This method [11] employs a deep neural model to identify out-of-distribution samples of image data, given only the examples from the normal class. A series of geometric transformations are applied to the normal class to create a multi-class dataset. A deep neural net, trained using this dataset, is then employed to discriminate the transformations applied. Subsequently, given an unseen instance, the model applies each transformation on it and assigns membership scores. The final normality score is determined based on the combined log-likelihood of softmax response vectors.

**4.2. Datasets**

We employ five benchmark datasets, namely, KDDCUP, MNIST, CIFAR-10, CatVsDog and UCF-Crime, to evaluate our proposed method, together with other methods described above.

- **KDDCUP:** The KDDCUP network intrusion dataset [18] contains samples of 41 dimensions. Similar to [35], categorical features are prepared by applying one-hot encoding. 20% of the "normal" samples form the minority group, while the rest 80% are treated as "attackers". As "normal" samples are the minorities, they are treated as anomalies.

- **MNIST:** The MNIST dataset [17] consists of 60,000 gray-scale 28×28 images of handwritten digits from 0 to 9. We formulate an anomaly detection task as per described in [5] and [34], where 4,859 images of digit 4 are randomly sampled as normal instances and 265 images are evenly sampled from all other categories as anomalies.

- **CIFAR-10:** The CIFAR-10 dataset [15] contains 60,000 color images of size 32×32 from 10 classes. We formulate an anomaly detection task with 5050 examples from class airplane (category 0) being the normal group and 450 images evenly sampled from the rest of the categories as anomalous instances.

- **CatVsDog:** The CatVsDog dataset consists of dogs and cats images of varying sizes, which are extracted from the ASIRRA dataset [8] following the settings specified in [11]. 12,500 images of dogs and 2,500 images of cats are sampled to form an anomaly detection task. The cat images are treated as anomalies.

- **UCF-Crime:** The UCF-Crime dataset [27] contains 1,900 long and untrimmed videos captured from CCTV cameras. It covers 13 categories of real-world crimes under diverse conditions, e.g., indoor and outdoor, day and night times. Example crime categories include fighting, burglary, etc. In both the training and testing sets, videos are of different lengths and anomalies happens at various temporal locations. Videos within the same category may contain diverse background scenes. Some of the videos may have multiple anomaly events.

**4.3. Evaluations**

We adopt Area Under the curve of the Receiver Operating Characteristic (AUROC) as the main evaluation metric to measure the discrimination power of different models. AUROC is a standard method to assess the effectiveness of a classifier [10]. It can be interpreted as the probability that an anomalous instance is assigned to a higher anomaly score than a normal instance [7]. In this section, we compare the performance of our method against other baseline methods.

**KDDCUP: Network Intrusion Data** In this experiment, we divide the KDDCUP dataset following the setting in [35]. 50% of the data is reserved for testing by random sampling. From the remaining 50% of the data reserved for training, we take all samples from the normal class and mix them with different percentages of samples from the anomaly class to form the training set. Parameters for this experiment (see Algorithm 1) are set to: \( p_0 = 35\% \), \( p = 30\% \), \( r = 10 \).

Table 2 and 3 reports the AUROC and AUPRC of OC-SVM, DAGMM and our model on the KDDCUP dataset after 200 epochs, with anomaly percentage in training set being 5%, 10% and 20%, respectively. It can be observed that an increase in the percentage of anomalous data undermines the detection performance of OC-SVM and DAGMM more severely, while our method remains robust to such changes.

Figure 4 shows the Receiver Operating Characteristic (ROC) curves of different models when the anomaly percentage of the training data is 20%. In our unsupervised setting where no prior knowledge of normal class is known, our method is clearly more robust to contaminated training data.

**Image Data** In table 4, we compare the AUROC scores obtained from OC-NN, OC-SVM, DAGMM, Geometric Transformation and our model, based on multiple image datasets. It should be noted that Geometric Transformation approach trains on data from the normal class only (hence classified as semi-supervised). Our method, on the other hand, does not require label information. Unless otherwise specified, we use NetVLAD [2] as feature representation for all the image datasets.

The parameters used (refer to Algorithm 1) for each image dataset are as follows: MNIST \( (p_0 = 25\%, p = 20\%, r = 5) \), CIFAR-10 \( (p_0 = 15\%, p = 10\%, r = 5) \) and CatVsDog \( (p_0 = 25\%, p = 20\%, r = 10) \). Detailed parameters for the experiments are presented in the appendix.
Table 1: Summary statistics of datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Normal Class</th>
<th>Input Dimension</th>
<th># Instances</th>
<th>Anomaly Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDDcup attack</td>
<td></td>
<td>1×120</td>
<td>494,021</td>
<td>20</td>
</tr>
<tr>
<td>MNIST digit 4</td>
<td></td>
<td>28×28</td>
<td>5,124</td>
<td>5</td>
</tr>
<tr>
<td>CIFAR-10 airplane (category 0)</td>
<td></td>
<td>32×32</td>
<td>5,500</td>
<td>8</td>
</tr>
<tr>
<td>CatVsDog dog</td>
<td></td>
<td>128×128</td>
<td>15,000</td>
<td>17</td>
</tr>
<tr>
<td>UCF-Crime non-crime scenes</td>
<td></td>
<td>varying</td>
<td>dep. on video</td>
<td>&lt; 35</td>
</tr>
</tbody>
</table>

Table 2: AUROC (in %) of different models with different anomaly percentage based on KDDCUP dataset. Our proposed method is much more immune to increase in anomaly percentage.

<table>
<thead>
<tr>
<th>Anomaly Percentage (%)</th>
<th>OC-SVM</th>
<th>DAGMM</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>96.8 ± 0.5</td>
<td>96.6 ± 1.1</td>
<td>98.2 ± 1.0</td>
</tr>
<tr>
<td>10</td>
<td>89.7 ± 0.1</td>
<td>88.6 ± 2.0</td>
<td>98.4 ± 0.8</td>
</tr>
<tr>
<td>20</td>
<td>61.6 ± 0.1</td>
<td>79.5 ± 2.0</td>
<td>93.5 ± 1.1</td>
</tr>
</tbody>
</table>

Table 3: AUPRC (in %) of different models with different anomaly percentage based on KDDCUP dataset. Our proposed method is much more immune to increase in anomaly percentage.

<table>
<thead>
<tr>
<th>Anomaly Percentage (%)</th>
<th>OC-SVM</th>
<th>DAGMM</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>77.8 ± 0.1</td>
<td>75.4 ± 0.7</td>
<td>94.5 ± 0.1</td>
</tr>
<tr>
<td>10</td>
<td>68.1 ± 0.1</td>
<td>53.4 ± 2.7</td>
<td>93.9 ± 0.1</td>
</tr>
<tr>
<td>20</td>
<td>45.4 ± 0.0</td>
<td>40.7 ± 2.8</td>
<td>90.3 ± 0.5</td>
</tr>
</tbody>
</table>

Figure 4: ROC comparison of our proposed method, DAGMM, and OC-SVM. Results are obtained based on the KDDCUP dataset, with 20% anomaly.

Results in Table 4 demonstrate an outstanding performance of our method over other unsupervised approaches. In addition, on CIFAR-10, the performance of our proposed algorithm is comparable to that of Geometric Transformation, a semi-supervised method. In the last column of Table 4, we report results obtained using distribution clustering alone. The combination of distribution clustering and autoencoder significantly improves discrimination against anomalies. Details of the network parameters used in the experiments are reported in the supplementary material.

We make several key observations based on the results in Table 4. Unlike all other methods that tend to perform better on simpler datasets (e.g. MNIST), the advantage of our method becomes more evident on datasets with higher complexity. Notably, our method outperforms other unsupervised approaches on CIFAR-10 and CatVsDog. The shortfall of our method on MNIST dataset could be due to the adoption of NetVLAD feature extractor (4096-d feature vectors), which may not be an ideal choice of feature representation since the images are pre-aligned and hence of low dimensionality. We repeat the same experiment using raw image pixel values from MNIST. It shows improved AUROC score, suggesting that raw feature is a better representation.

While our method is able to produce results comparable to semi-supervised approaches, the gap is wider on CatVsDog dataset as compared to CIFAR-10. We attribute this to the high noise level in the CatVsDog dataset. For example, some images consist of both dog and cats. Moreover, training on normal data (with augmentation through geometric transformation) gives Geometric Transform a natural advantage. According to [8], ASIRRA dataset, from which the CatVsDog is extracted, is deemed extremely challenging for computers. Sample images of the dataset are presented in the supplementary material.

**Video Data** We apply the proposed approach on UCF-Crime dataset [27], with features extracted using C3D [28] descriptor. In default C3D settings, every 16-frame segment is aggregated to generate 1 feature vector.

In [27], although the AUROC score of each video category is not reported, the AUROC averaged across the entire test set of UCF-Crime dataset is 75.4%. It is achieved by adopting a weakly-supervised method called multiple instance learning (MIL). We applied our method on each crime category by combining all test videos in each category as a single dataset for training. The results are tabulated in Table 5. Our method is able to score 72.9%, losing by just a small margin of 2.5% to [27]. We note that crimes...
Table 4: AUROC in %. Highest score among all methods and highest score among all unsupervised methods are highlighted. On complex datasets such as CIFAR-10 and CatVsDog, our proposed method has higher performance gain among all unsupervised methods. The last column presents results obtained using distribution clustering alone.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>70.0</td>
<td>98.2</td>
<td>50.3</td>
<td>90.2</td>
<td>82.4 ± 1.8 (raw) / 70.5 ± 2.1 (NetVLAD)</td>
<td>63.3</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>63.8</td>
<td>73.3</td>
<td>49.0</td>
<td>69.7</td>
<td>73.6 ± 0.6</td>
<td>48.7</td>
</tr>
<tr>
<td>CatVsDog</td>
<td>50.8</td>
<td>88.3</td>
<td>43.4</td>
<td>56.2</td>
<td>78.0 ± 1.2</td>
<td>56.1</td>
</tr>
</tbody>
</table>

Table 5: AUROC for each crime scene category UCF-Crime.

<table>
<thead>
<tr>
<th>Crime Scene</th>
<th>No. of videos</th>
<th>AUROC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abuse</td>
<td>2</td>
<td>66.4</td>
</tr>
<tr>
<td>Arrest</td>
<td>5</td>
<td>63.2</td>
</tr>
<tr>
<td>Arson</td>
<td>9</td>
<td>65.6</td>
</tr>
<tr>
<td>Assault</td>
<td>3</td>
<td>76.9</td>
</tr>
<tr>
<td>Burglary</td>
<td>13</td>
<td>76.8</td>
</tr>
<tr>
<td>Explosion</td>
<td>21</td>
<td>72.8</td>
</tr>
<tr>
<td>Fighting</td>
<td>5</td>
<td>76.4</td>
</tr>
<tr>
<td>Road Accidents</td>
<td>23</td>
<td>79.5</td>
</tr>
<tr>
<td>Robbery</td>
<td>5</td>
<td>78.1</td>
</tr>
<tr>
<td>Shooting</td>
<td>23</td>
<td>73.3</td>
</tr>
<tr>
<td>Shoplifting</td>
<td>21</td>
<td>63.5</td>
</tr>
<tr>
<td>Stealing</td>
<td>5</td>
<td>73.3</td>
</tr>
<tr>
<td>Vandalism</td>
<td>5</td>
<td>83.5</td>
</tr>
</tbody>
</table>

Average: 10.77 | Ave. over all categories: 73.0

Average: 72.9

Table 6: Performance comparison based on AUROC scores on selected videos with DAGMM [35]

<table>
<thead>
<tr>
<th>Crime scene</th>
<th># Video selected</th>
<th>Ours</th>
<th>DAGMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrest</td>
<td>2</td>
<td>70.6</td>
<td>52.2</td>
</tr>
<tr>
<td>Arson</td>
<td>3</td>
<td>67.8</td>
<td>60.1</td>
</tr>
<tr>
<td>Burglary</td>
<td>4</td>
<td>79.2</td>
<td>67.4</td>
</tr>
<tr>
<td>Fighting</td>
<td>3</td>
<td>77.1</td>
<td>57.0</td>
</tr>
</tbody>
</table>

Run Time: Excluding feature extraction and clustering process, on a single NVIDIA Tesla P100 GPU, our method takes 4 minutes 20 seconds on average to complete the CIFAR-10 experiment described above (consisting of 5,500 instances). This timing is averaged over 5 runs.

4.4. Ablation Study

Initialization: To examine the effect of adopting distribution clustering as the initialization method for high-dimensional data, a variety of other mainstream clustering methods, including K-means, HDBSCAN [21] and Gaussian Mixture Model (GMM) are used to replace the distribution clustering component in the initial normal subset selection. We compare results on the CIFAR-10 task.

For K-means and GMM, the number of clusters/components is set to 20, which is consistent with the setting of the GMM employed in the proposed model. For HDBSCAN, the minimum size of a cluster is set to 5, that follows the setting as distribution clustering. Table 8 reports the AUROC scores obtained from CIFAR-10 anomaly detection task with different clustering techniques. The results demonstrate that using distribution clustering initialization provides better supervisory signals and leads to favorable performance.

To further understand the effectiveness of distribution clustering, we tabulated the AUROC achieved using distribution clustering alone, for the experiments on image data (refer to right-most column of Table 4). Surprisingly, this...
Figure 5: Anomaly scores (normalized) plotted against ground truth (flagged by orange lines). Compared to DAGMM, our method shows much better correspondence to the ground truth.

Table 7: AUROC score when varying anomaly percentage $p$. Ground truth anomaly percentages are shown on the left column.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>G.T.</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST (raw)</td>
<td>5</td>
<td>42.7 ± 1.0</td>
<td>55.8 ± 1.4</td>
<td>63.0 ± 1.3</td>
<td>70.5 ± 2.1</td>
<td>70.3 ± 0.8</td>
<td>70.2 ± 1.5</td>
<td>71.2 ± 1.0</td>
<td>73.7 ± 0.3</td>
<td>70.0 ± 1.9</td>
<td>71.1 ± 1.5</td>
</tr>
<tr>
<td>Cifar10</td>
<td>8</td>
<td>72.8 ± 0.7</td>
<td>73.6 ± 0.6</td>
<td>71.9 ± 0.7</td>
<td>70.9 ± 0.5</td>
<td>67.5 ± 0.5</td>
<td>71.6 ± 0.4</td>
<td>69.0 ± 0.3</td>
<td>68.8 ± 1.0</td>
<td>66.0 ± 1.0</td>
<td>64.0 ± 0.9</td>
</tr>
<tr>
<td>CatVsDog</td>
<td>17</td>
<td>60.4 ± 1.7</td>
<td>66.5 ± 2.1</td>
<td>74.0 ± 4.4</td>
<td>78.0 ± 1.2</td>
<td>74.0 ± 2.5</td>
<td>70.1 ± 3.2</td>
<td>69.5 ± 2.7</td>
<td>62.4 ± 4.0</td>
<td>60.3 ± 1.6</td>
<td>61.0 ± 3.3</td>
</tr>
</tbody>
</table>

Influence of Anomaly Percentage $p$ is the assumed anomaly percentage that serves as a threshold for normal candidate selection from the clustering output. Table 7 shows the AUROC scores with varying $p$ values. We observe that, except for the MNIST dataset, a small overestimation above the ground truth values have little impact on performance. AUROC scores degrade gracefully as $p$ increases to be more than 10% above the ground truth. This implies the autoencoder was still able to generalize well on the “unseen” normal data that was discarded.

5. Discussion and Conclusion

This paper presents an end-to-end method for anomaly detection under a fully unsupervised setting. The key insight of our algorithm is to model normal data. We first leverage distribution clustering technique to make an educated guess on the normal data. By incorporating clustering to provide supervisory signals, we iterate between hypothesizing normal candidate subset and representation learning. This framework iteratively distills out anomalous data and improves the learned representation of normal data. Extensive experiments on benchmark datasets demonstrate our proposed method outperforms existing unsupervised approaches and is comparable to semi-supervised solutions in most cases.

Limitations and future work: Using only an autoencoder may be insufficient to handle highly complex patterns and hence falls short on difficult dataset such as CatVsDog. For future work, we seek to explore more sophisticated generative frameworks for representation learning.

Acknowledgment This research is supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant.
References


