Memorizing Normality to Detect Anomaly: Memory-augmented Deep Autoencoder for Unsupervised Anomaly Detection

Dong Gong[1], Lingqiao Liu[1], Vuong Le[2], Budhaditya Saha[2],
Moussa Reda Mansour[3], Svetha Venkatesh[2], Anton van den Hengel[1]
[1]The University of Adelaide, Australia [2]A2I2, Deakin University [3]University of Western Australia
https://donggong1.github.io/anomdec-memae

Abstract

Deep autoencoder has been extensively used for anomaly detection. Training on the normal data, the autoencoder is expected to produce higher reconstruction error for the abnormal inputs than the normal ones, which is adopted as a criterion for identifying anomalies. However, this assumption does not always hold in practice. It has been observed that sometimes the autoencoder "generalizes" so well that it can also reconstruct anomalies well, leading to the miss detection of anomalies. To mitigate this drawback for autoencoder based anomaly detector, we propose to augment the autoencoder with a memory module and develop an improved autoencoder called memory-augmented autoencoder, i.e. MemAE. Given an input, MemAE firstly obtains the encoding from the encoder and then uses it as a query to retrieve the most relevant memory items for reconstruction. At the training stage, the memory contents are updated and are encouraged to represent the prototypical elements of the normal data. At the test stage, the learned memory will be fixed, and the reconstruction is obtained from a few selected memory records of the normal data. The reconstruction will thus tend to be close to a normal sample. Thus the reconstructed errors on anomalies will be strengthened for anomaly detection. MemAE is free of assumptions on the data type and thus general to be applied to different tasks. Experiments on various datasets prove the excellent generalization and high effectiveness of the proposed MemAE.

1. Introduction

Anomaly detection is an essential task with critical applications in various areas, such as video surveillance [26]. The unsupervised anomaly detection [47, 43, 48, 32, 7] is to learn a normal profile given only the normal data examples and then identify the samples not conforming to the normal profile as anomalies, which is challenging due to

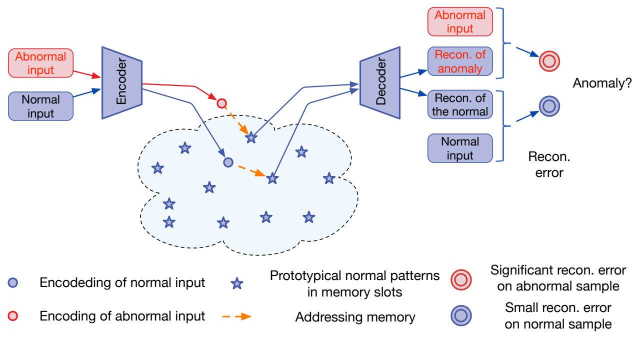

Abnormal input
Normal input
Encoder
Decoder
Abnormal input
Recon. of anomaly
Recon. of the normal
Normal input
Anomaly?
Recon. error
Encodeding of normal input
Encoding of abnormal input
Prototypical normal patterns in memory slots
Addressing memory
Significant recon. error on abnormal sample
Small recon. error on normal sample

Figure 1. Anomaly detection via the proposed MemAE. After training on the dataset only with normal samples, the memory in MemAE records the prototypical normal patterns. Given an *abnormal* input, MemAE retrieves the most relevant *normal* patterns in memory for reconstruction, resulting in an output significantly different to the abnormal input. To simplify the visualization, we assume only one memory item is addressed here.

the lack of human supervision. Notably, the problem becomes even more difficult when the data points lay in a high-dimensional space (*i.e.* videos), since modeling the high-dimensional data is notoriously challenging [47].

Deep autoencoder (AE) [2, 18] is a powerful tool to model the high-dimensional data in the unsupervised setting. It consists of an encoder to obtain a compressed encoding from the input and a decoder that can reconstruct the data from the encoding. The encoding essentially acts as an information bottleneck which forces the network to extract the typical patterns of high-dimensional data. In the context of anomaly detection, the AE is usually trained by minimizing the reconstruction error on the normal data and then uses the reconstruction error as an indicator of anomalies. It is generally assumed [48, 11, 45] that the reconstruction error will be lower for the normal input since they are close to the training data, while the reconstruction error becomes higher for the abnormal input.

However, this assumption may not always hold, and sometimes the AE can "generalize" so well that it can also reconstruct the abnormal inputs well. This observation has been made in the existing literature [48, Figure 1] and also in this paper (See Figure 4 and 6). The assumption that anomaly incurs higher reconstruction error might be somehow questionable since there are no training samples for anomalies and the reconstruction behavior for anomaly inputs should be unpredictable. If some anomalies share common compositional patterns (*e.g.* local edges in images) with the normal training data or the decoder is "too strong" for decoding some abnormal encodings well, AE is very likely to reconstruct the anomalies well.

To mitigate the drawback of AEs, we propose to augment the deep autoencoder with a memory module and introduce a new model memory-augmented autoencoder, *i.e.* MemAE. Given an input, MemAE does not directly feed its encoding into the decoder but uses it as a query to retrieve the most relevant items in the memory. Those items are then aggregated and delivered to the decoder. Specifically, the above process is realized by using attention based memory addressing. We further propose to use a differentiable hard shrinkage operator to induce sparsity of the memory addressing weights, which implicitly encourage the memory items to be close to the query in the feature space.

In the training phase of MemAE, we update the memory content together with the encoder and decoder. Due to the sparse addressing strategy, the MemAE model is encouraged to optimally and efficient use the limited number of memory slots, making the memory to record the prototypical normal patterns in the normal training data to obtain low average reconstruction error (See Figure 3). In the test phase, the learned memory content is fixed, and the reconstruction will be obtained by using a small number of the *normal memory items*, which are selected as the neighborhoods of the encoding of the input. Because the reconstruction is obtained normal patterns in memory, it tends to be close to the normal data. Consequently, the reconstruction error tends to be highlighted if the input is not similar to normal data, that is, an anomaly. The schematic illustration is shown in Figure 1. The proposed MemAE is free of the assumption on the types of data and thus can be generally applied to solve different tasks. We apply the proposed MemAE on various public anomaly detection datasets from different applications. Extensive experiments prove the excellent generalization and high effectiveness of MemAE.

2. Related Work

Anomaly detection In unsupervised anomaly detection, only normal samples are available as training data [4]. A natural choice for handling the problem is thus the one-class classification methods, such as one-class SVM [5, 34] and deep one-class networks [31, 3], which seeks to learn a discriminative hyperplane surrounding the normal samples. Unsupervised clustering methods, such as the k-means method and Gaussian Mixture Models (GMM) [47, 40], have also been applied to build a detailed profile of the normal data for identifying the anomalies. These methods usually suffer from suboptimal performance when processing high-dimensional data.

Reconstruction-based methods are proposed relying on an assumption that the anomalies cannot be represented and reconstructed accurately by a model learned only on normal data [48]. Different techniques, such as PCA methods [14, 15] and sparse representation [25, 45], have been used to learn the representation of the normal patterns. Specifically, sparse representation methods [25, 45] jointly learn a dictionary and the sparse representation of the normal data for detecting anomalies. The restricted feature representations limit the performances. Some very recent works [43, 46, 48, 6] train deep autoencoders for anomaly detection. For example, structured energy based deep neural network [43] is used to model the training samples. Zong *et al.* [48] proposed to jointly model the encoded features and the reconstruction error in a deep autoencoder. Although the reconstruction based methods have achieved fruitful results, their performances are restricted by the under-designed representation of the latent space.

Due to the critical application scenario, a series of methods are specifically designed for *video anomaly detection* [24, 44, 11, 21]. Kim and Grauman [15] use a mixture of probabilistic PCA (MPPCA) to model optical flow features. Mahadevan *et al.* [27] model the video via a mixture of dynamic textures (MDT). Lu *et al.* [25] proposed an efficient sparse coding-based method with multiple dictionaries. Zhao *et al.* [44] update the dictionary in an online manner. Deep learning based methods [11, 26, 24, 32] are proposed to use the information in both the spatial and temporal domain. Hasan *et al.* [11] detect the anomalies according to the reconstruction error of a convolutional AE. Zhao *et al.* [45] proposed to use 3D convolution based reconstruction and prediction. Luo *et al.* [26] iteratively update the sparse coefficients via a stacked RNN to detect anomalies in videos. Liu *et al.* [24] train a frame prediction network by incorporating different techniques including gradient loss, optical flow, and adversarial training. However, these methods lack a reliable mechanism to encourage the model to induce large reconstruction error on the anomalies.

Memory networks Memory-augmented networks have attracted increasing interest for solving different problems [10, 39, 33]. Graves *et al.* [10] use external memory to extend the capability of neural networks, in which content-based attention is used for addressing the memory. Considering that memory can record information stably, Santoro *et al.* [33] use a memory network to handle the one-shot learning problem. The external memory has also been used for multi-modal data generation [16, 22], for circumventing the mode collapse issue and preserving detailed data structure.

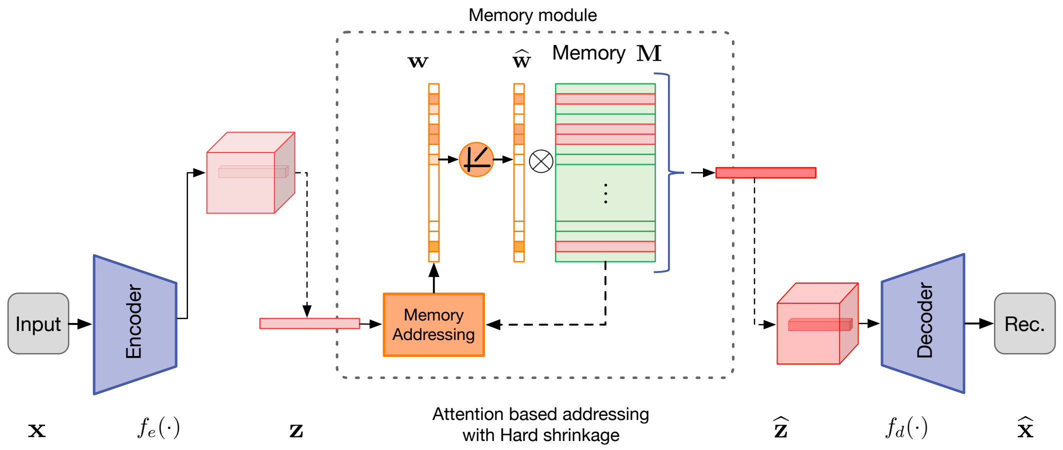


Figure 2. Diagram of the proposed MemAE. The memory addressing unit takes the encoding $\mathbf{z}$ as query to obtain the soft addressing weights. The memory slots can be used to model the whole encoding or the features on one pixel of the encoding (as shown in the figure). Note that $\widehat{\mathbf{w}}$ is normalized after the hard shrinkage operation.

3. Memory-augmented Autoencoder

3.1. Overview

The proposed MemAE model consists of three major components - an encoder (for encoding input and generating query), a decoder (for reconstruction) and a memory module (with a memory and the associated memory addressing operator). As shown in Figure 2, given an input, the encoder first obtains the encoding of the input. By using the encoded representation as a query, the memory module retrieves the most relevant items in the memory via the attention-based addressing operator, which are then delivered to the decoder for reconstruction. During training, the encoder and decoder are optimized to minimize the reconstruction error. The memory contents are simultaneously updated to record the prototypical elements of the encoded normal data. Given a testing sample, the model performs reconstruction merely using a restricted number of the normal patterns recorded in the memory. As a result, the reconstruction tends to be close to the normal sample, resulting in small reconstruction errors for normal samples and large errors on anomalies, which will be used as a criterion to detect the anomalies.

3.2. Encoder and Decoder

The encoder is used to represent the input in an informative latent domain. The encoded representation performs as a query to retrieve the relevant items in the memory. In our model, the encoder can be seen as a *query generator*. The decoder is trained to reconstruct the samples by taking the retrieved memories as input.

We first define $\mathbb{X}$ to represent the domain of the data samples and $\mathbb{Z}$ to represent the domain of the encodings. Let $f_e(\cdot) : \mathbb{X} \to \mathbb{Z}$ denote the encoder and $f_d(\cdot) : \mathbb{Z} \to \mathbb{X}$ denote the decoder. Given a sample $\mathbf{x} \in \mathbb{X}$, the encoder converts it to an encoded representation as $\mathbf{z} \in \mathbb{Z}$; and the decoder is trained to reversely mapping a latent representation $\widehat{\mathbf{z}} \in \mathbb{Z}$ to the domain $\mathbb{X}$ as follows

$$\mathbf{z} = f_e(\mathbf{x};\ \theta_e), \tag{1}$$

$$\widehat{\mathbf{x}} = f_d(\widehat{\mathbf{z}};\ \theta_d), \tag{2}$$

where θ_e and θ_d denote the parameters of the encoder $f_e(\cdot)$ and decoder $f_d(\cdot)$, respectively. In the proposed MemAE, $\mathbf{z}$ is used to retrieve the relevant memory items; and $\widehat{\mathbf{z}}$ is obtained using the retrieved items. For the standard AE model, there is $\widehat{\mathbf{z}} = \mathbf{z}$. Our method is agnostic to the structures of the encoder and decoder, which can be specially selected for different applications.

In testing, given a sample $\mathbf{x}$, we use the ℓ_2-norm based mean square error (MSE), *i.e.* $e = \|\mathbf{x} - \widehat{\mathbf{x}}\|_2^2$, to measure of the reconstruction quality, which is used as the criterion for anomaly detection.

3.3. Memory Module with Attention-based Sparse Addressing

The proposed memory module consists of a memory to record the prototypical encoded patterns and an attention-based addressing operator for accessing the memory.

3.3.1 Memory-based Representation

The memory is designed as a matrix $\mathbf{M} \in \mathbb{R}^{N \times C}$ containing N real-valued vectors of fixed dimension C. For convenience, we assume C is same to the dimension of $\mathbf{z}$ and let $\mathbb{Z} = \mathbb{R}^C$. Let the row vector $\mathbf{m}_i, \forall i \in [N]$ denote the i-th row of $\mathbf{M}$, where $[N]$ denotes the set of integers from 1 to N. Each $\mathbf{m}_i$ denotes a memory item. Given a query (*i.e.*

encoding) $\mathbf{z} \in \mathbb{R}^C$, the memory network obtains $\widehat{\mathbf{z}}$ relying a soft *addressing vector* $\mathbf{w} \in \mathbb{R}^{1\times N}$ as follows

$$\widehat{\mathbf{z}} = \mathbf{w}\mathbf{M} = \sum\nolimits_{i=1}^{N} w_i \mathbf{m}_i, \tag{3}$$

where $\mathbf{w}$ is a row vector with non-negative entries that sum to one and w_i denotes the i-th entry of $\mathbf{w}$. The weight vector $\mathbf{w}$ is computed according to $\mathbf{z}$. As shown in Eq. (3), the addressing weight $\mathbf{w}$ is required for accessing the memory. The hyper-parameter N defines the maximum capacity of the memory. Although it is non-trivial to find the optimal N for different datasets, MemAE is insensitive to the setting of N, fortunately (See Section 4.2). A large enough N can work well for each dataset.

3.3.2 Attention for Memory Addressing

In MemAE, the memory $\mathbf{M}$ is designed to explicitly record the prototypical normal patterns during training. We define the memory as a content addressable memory [39, 30] with an addressing scheme that computes attention weights $\mathbf{w}$ based on the similarity of the memory items and the query $\mathbf{z}$. As visualized in Figure 1, we compute each wight w_i via a softmax operation:

$$w_i = \frac{\exp(d(\mathbf{z}, \mathbf{m}_i))}{\sum_{j=1}^{N}\exp(d(\mathbf{z}, \mathbf{m}_j))}, \tag{4}$$

where $d(\cdot,\cdot)$ denotes a similarity measurement. Similar to [33], we define $d(\cdot,\cdot)$ as cosine similarity:

$$d(\mathbf{z}, \mathbf{m}_i) = \frac{\mathbf{z}\mathbf{m}_i^\top}{\|\mathbf{z}\|\|\mathbf{m}_i\|}. \tag{5}$$

As shown in Eq. (3), (4) and (5), the memory module retrieves the memory items most similar to $\mathbf{z}$ to obtain the representation $\widehat{\mathbf{z}}$. Due to the restricted memory size and the sparse addressing technique (introduced in Section 3.3.3), only a small number of memory items can be addressed every time. Accordingly, the beneficial behaviors of the memory module can be interpreted as follows.

In *training phase*, the decoder in MemAE is restricted to perform reconstruction merely using a very small number of addressed memory items, rendering the requirement for efficient utilization of the memory items. The reconstruction supervision thus forces the memory to record the most representative prototypical patterns in the input normal patterns. In Figure 3, we visualize the trained single memory slots, which shows that each single memory slot records the prototypical normal patterns in the training data.

In *testing phase*, given the trained memory, only the normal patterns in the memory can be retrieved for reconstruction. Thus the normal samples can naturally be reconstructed well. Conversely, the encoding of an abnormal input will be replaced by the retrieved normal patterns, resulting in significant reconstruction errors on anomalies (See visualized examples in Figure 4).

3.3.3 Hard Shrinkage for Sparse Addressing

As discussed above, performing reconstruction with a restricted number of normal patterns in the memory helps to induce large reconstruction error on anomalies. The attention-based addressing tends to approach this naturally [10]. However, some anomalies may still have the chance to be reconstructed well with a complex combination of the memory items via a dense $\mathbf{w}$ containing many small elements. To alleviate this issue, we apply a hard shrinkage operation to promote the sparsity of $\mathbf{w}$:

$$\widehat{w}_i = h(w_i; \lambda) = \begin{cases} w_i, & \text{if } w_i > \lambda, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

where $\widehat{w}_i$ denotes the i-th entry of the memory addressing weight vector $\widehat{\mathbf{w}}$ after shrinkage and λ denotes the shrinkage threshold. It is not easy to directly implement the backward of the discontinuous function in Eq. (6). For simplicity, considering that all entries in $\mathbf{w}$ are non-negative, we rewrite the hard shrinkage operation using the continuous ReLU activation function as

$$\widehat{w}_i = \frac{\max(w_i - \lambda, 0)\cdot w_i}{|w_i - \lambda| + \epsilon}, \tag{7}$$

where $\max(\cdot, 0)$ is also known as ReLU activation, and ϵ is a very small positive scalar. In practice, setting the threshold λ as a value in the interval $[1/N, 3/N]$ can render desirable results. After the shrinkage, we re-normalize $\widehat{\mathbf{w}}$ by letting $\widehat{w}_i = \widehat{w}_i / \|\widehat{\mathbf{w}}\|_1, \forall i$. The latent representation $\widehat{\mathbf{z}}$ will be obtained via $\widehat{\mathbf{z}} = \widehat{\mathbf{w}}\mathbf{M}$.

The sparse addressing encourages the model to represent an example using fewer but more relevant memory items, leading to learning more informative representations in memory. In addition, similar to the sparse representation methods [44], encouraging sparsity of the addressing weights is beneficial in testing due to that the memory $\mathbf{M}$ is trained to adapt the sparse $\mathbf{w}$. Encouraging sparsity in $\mathbf{w}$ will also alleviate the issue that an abnormal sample may be fairly reconstructed well with dense addressing weights. Comparing with the sparse representation methods [44, 26], the proposed method obtains the desired sparse $\mathbf{w}$ via once efficient forward operation, instead of the iterative updating.

3.4. Training

Given a dataset $\{\mathbf{x}^t\}_{t=1}^{T}$ containing T samples, let $\widehat{\mathbf{x}}^t$ denote the reconstructed sample corresponding the each training sample $\mathbf{x}^t$. We firstly conduct to minimize the reconstruction error on each sample:

$$R(\mathbf{x}^t,\ \widehat{\mathbf{x}}^t) = \|\mathbf{x}^t - \widehat{\mathbf{x}}^t\|_2^2, \tag{8}$$

where the ℓ_2-norm is used to measure the reconstruction error. Let $\widehat{\mathbf{w}}^t$ denote the memory addressing weights for each sample $\mathbf{x}^t$. To further promote the sparsity of $\widehat{\mathbf{w}}$, in addition to the shrinkage operation in Eq. (7), we minimize a sparsity regularizer on $\widehat{\mathbf{w}}$ during training. Considering that all entries of $\widehat{\mathbf{w}}$ are non-negative and $\|\widehat{\mathbf{w}}\|_1 = 1$, instead of minimizing $\|\widehat{\mathbf{w}}\|_1$ [8, 9], we minimize the entropy of $\widehat{\mathbf{w}}^t$:

$$E(\widehat{\mathbf{w}}^t) = \sum\nolimits_{i=1}^{T} -\widehat{w}_i \cdot \log(\widehat{w}_i). \tag{9}$$

The hard shrinkage operation in Eq. (7) and the entropy loss Eq. (9) jointly promote the sparsity of the generated addressing weights. More detailed ablation studies and discussions can be found in Section 4.4.

By combining loss functions in Eq. (8) and (9), we construct the training objective for MemAE as:

$$L(\theta_e, \theta_d, \mathbf{M}) = \frac{1}{T} \sum\nolimits_{t=1}^{T} \left(R(\mathbf{x}^t,\ \widehat{\mathbf{x}}^t) + \alpha E(\widehat{\mathbf{w}}^t) \right), \tag{10}$$

where α is a hyper-parameter in training. In practice, $\alpha = 0.0002$ leads to desirable results in all our experiments. During training, the memory $\mathbf{M}$ is updated through optimization via backpropagation and gradient descent. In backward pass, only the gradients for the memory items with non-zero addressing weights w_i can be non-zero.

4. Experiments

In this section, we validate the proposed MemAE for anomaly detection. To show the generality and applicability of the proposed model, we conduct experiments on five datasets of three different tasks. The results are compared with different baseline models and state-of-the-art techniques. The proposed MemAE is applied to all datasets following previous sections. MemAE and its variants are implemented using PyTorch [29] and trained using the optimizer Adam [17] with a learning rate of 0.0001. We make them and other encoder-decoder models such as VAE to have the similar model capacity.

4.1. Experiments on Image Data

We first conduct the experiments to detect outliers image [32] and evaluate the performance on two image datasets: MNIST [20] and CIFAR-10 [19], both of which contain images belonging to 10 classes. For each dataset, we construct 10 anomaly detection (*i.e.* one-class classification) datasets by sampling images from each class as normal samples and sampling anomalies from the rest classes. The normal datums are split into training and testing set with a rate of 2:1. Following the setting used in [43, 48], the training set only consists of normal samples and has no overlapping with the testing set. The anomaly proposition is controlled around 30%. 10% of the original training data is left for validation.

In this experiment, we focus on validating the proposed memory module and implement the encoder and decoder as plain convolutional neural networks. We first define Conv2(k, s, c) to denote a 2D convolution layer, where k, s and c are the kernel size, stride size and the number of channels, respectively. For MNIST, we implement the encoder using three convolution layers: Conv2(3, 2, 32)-Conv2(3, 2, 16)-Conv2(3, 3, 8). The decoder is implemented as Dconv2(3, 3, 16)-Dconv2(3, 2, 32)-Dconv2(3, 2, 1), where Dconv2 denotes the 2D deconvolution layer. Except for the last Dconv2, each layer is followed by a batch normalization (BN) [13] and a leaky ReLU activation. Such design is applied for all datasets in the following. Considering the higher data complexity of CIFAR-10, we use the encoder and decoder with larger capacities: Conv2(3, 2, 64)-Conv2(3, 2, 128)-Conv2(3, 2, 128)-Conv2(3, 2, 256) and Dconv2(3, 2, 128)-Dconv2(3, 2, 128)-Dconv2(3, 2, 64)-Dconv2(3, 2, 3). We process the MNIST and CIFAR-10 datasets as gray images and RGB images, respectively. Memory sizes N for MNIST and CIFAR-10 are set as 100 and 500, respectively.

We compare the proposed model with several conventional and deep learning based methods for general anomaly detection as baselines, including one-class SVM (OC-SVM) [35], kernel density estimation (KDE) [28], a deep variational autoencoder (VAE) [18], a deep autoregressive generative model PixCNN [38] and the deep structured energy-based model (DSEBM) [43]. Specifically, for the density estimation methods (*e.g.* KDE and PixCNN) and the reconstruction based methods (*e.g.* VAE and DSEBM), the log-likelihood and reconstruction error are used to calculate the regularity score, respectively. Note that for fair comparison with other methods, we calculate the regularity score of VAE based on only reconstruction error. We also conduct the comparisons with some baseline variants of MemAE to show the importance of the major components, including the antueocoder without memory module (AE) and a variant of MemAE without the sparse shrinkage and the entropy loss (MemAE-nonSpar). In all experiments, AE, MemAE-nonSpar, and VAE share a similar capacity with the full MemAE model by using the same encoder and decoder. In testing, we scale the reconstruction error to the range $[0, 1]$ as the criterion to identify the anomalies. Following [27, 26, 1], we use the AUC (Area Under Curve) as the measurement for performance evaluation, which is obtained by calculating the area under the Receiver Operation Characteristic (ROC) with a varying threshold. Table 1 shows the average AUC values on the 10 sampled datasets.

As shown in Table 1, the proposed MemAE generally outperforms the compared methods. The memory-augmented models significantly outperform the AE without memory. And the MemAE model with sparse addressing yields better results. Images in MNIST only contain simple patterns, *i.e.* digits, which is easy to model. VAE can thus produce satisfactory results by using a simple Gaussian distribution to model latent space. All methods perform better on MNIST than CIFAR-10, since the images in CIFAR-10 have more complex content and exhibit larger

Table 1. Experimental results on image data. Average AUC values on 10 anomaly detection datasets sampled from MNIST and CIFAR-10 are shown.

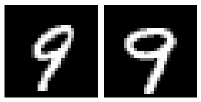
Dataset	MNIST	CIFAR-10
OC-SVM [35]	0.9499	0.5619
KDE	0.8116	0.5756
VAE [18]	0.9643	0.5725
PixCNN [38]	0.6141	0.5450
DSEBM [43]	0.9554	0.5725
AE	0.9619	0.5706
MemAE-nonSpar	0.9725	0.6058
MemAE	**0.9751**	**0.6088**

intra-class variance on several classes, which incurs unsatisfactory average ACU. Nevertheless, among the compared models with similar capacities, MemAE achieves superior performance than the competitors, which proves the effectiveness of the proposed memory module.

4.1.1 Visualizing How the Memory Works

Considering that the images in MNIST contain the patterns easy to identify, we use it to show how the proposed memory module works for anomaly detection.

What the memory learns. We first visualize what the memory learns from MNIST by randomly sampling a single memory slot and performing decoding on it. Figure 3 visualizes the memory learned on the MNIST digits "9" by treating them as normal samples. Since MemAE usually performs reconstruction via a combination of several addressed items, the decoded single slot appears blurry and noisy. Nevertheless, as shown in Figure 3(b), the memory slots record the different prototypical patterns of the normal training samples (*i.e.* digits "9").


(a) Training samples

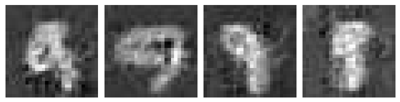
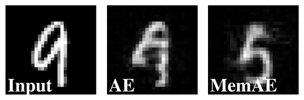
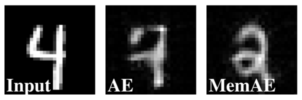

(b) Decoded single memory item

Figure 3. Visualization of the memory slots learned on MNIST by treating digits "9" as normal data. We randomly select a single memory item and perform decoding. The decoded single memory slot in (b) appears as a prototypical pattern of the normal samples.

How memory augments reconstruction. In Figure 4, we visualize the image reconstruction process under the memory augmentation. Since the trained memory only records the normal prototypical patterns, given an abnormal input "9", the MemAE trained on "5" reconstructs a "5", resulting in significant reconstruction error on the abnormal input. Note that the reconstructed "5" of MemAE has a similar shape of the input "9" since the memory module retrieves the most similar normal patterns. The AE model without memory tends to learn some representations more locally. Thus an abnormal sample may also be reconstructed well.

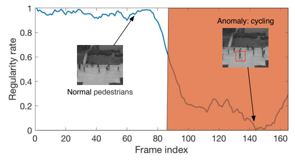


(a) Training on the normal "5"

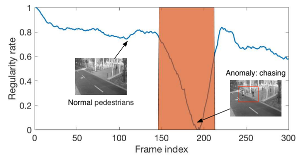


(b) Training on the normal "2"

Figure 4. Visualization of the reconstruction results of AE and MemAE on MNIST. (a) The models are trained on "5". The input is an image of "9". (b) The models are trained on "2". The input is an image of "4". The MemAE retrieves the normal memory items for reconstructions and obtains the results significantly different from the input anomalies.



(a) UCSD-Ped2



(b) ShanghaiTech

Figure 5. Normality scores of the video frames obtain by MemAE. The score decreases immediately when some anomalies appear in the video frame.

4.2. Experiments on Video Anomaly Detection

Anomaly detection on video aims to identify the unusual contents and moving patterns in the video, which is an essential task in video surveillance. We conduct experiments on three real-world video anomaly detection datasets, *i.e.* UCSD-Ped2 [27], CUHK Avenue [25] and ShanghaiTech [26]. Specifically, the most recent benchmark dataset ShanghaiTech contains more than 270,000 training frames and more than 42,000 frames (with about 17,000 abnormal frames) for testing, which covers 13 different scenes. In the datasets, objects except for pedestrians (*e.g.* vehicles) and strenuous motion (*e.g.* fighting and chasing) are treated as anomalies.

To preserve the video temporal information, we implement the encoder and decoder using 3D convolutions to extract the spatial-temporal features in video [36]. Accordingly, the input of the network is a cuboid constructed by stacking 16 neighbor frames in grayscale. The structures of encoder and decoder are designed as: Conv3(3, 2, 96)-Conv3(3, 2, 128)-Conv3(3, 2, 256)-Conv3(3, 2, 256) and Dconv3(3, 2, 256)-Dconv3(3, 2, 128)-Dconv3(3, 2, 96)-Dconv3(3, 2, 1), where Conv3 and Dconv3 denote 3D convolution and deconvolution, respectively. A BN and a leaky ReLU activation follow each layer (except the last one). We set $N = 2000$. Considering the complexity of the video data, we let each memory slot record the features on one pixel in the feature maps, corresponding to a sub-area of the video clip. The memory is thus a matrix of 2000×256. In testing, the normality of each frame is evaluated by the reconstruction error of the cuboid centering on it. Following [11, 26], we obtain the normality score p_u of the u-th frame by normalizing the errors to range $[0, 1]$:

Table 2. AUC of different methods on video datasets UCSD-Ped2, CUHK Avenue and ShanghaiTech.

	Method\Dataset	UCSD-Ped2	CUHK	SH.Tech
Non-Recon.	MPPCA [15]	0.693	-	-
	MPPCA+SFA [27]	0.613	-	-
	MDT [27]	0.829	-	-
	AMDN [41]	0.908	-	-
	Unmasking [37]	0.822	0.806	-
	MT-FRCN [12]	0.922	-	-
	Frame-Pred [26]	**0.954**	**0.849**	**0.728**
Recon.	AE-Conv2D [11]	0.850	0.800	0.609
	AE-Conv3D [45]	0.912	0.771	-
	TSC [26]	0.910	0.806	0.679
	StackRNN [26]	0.922	0.817	0.680
	AE	0.917	0.810	0.697
	MemAE-nonSpar	0.929	0.821	0.688
	MemAE	**0.941**	**0.833**	**0.712**

$$p_u = 1 - \frac{e_u - \min_u(e_u)}{\max_u(e_u) - \min_u(e_u)}, \tag{11}$$

where e_u denotes the reconstruction error the u-th frame in a video episode. The value of p_u closer to 0 indicates the frame is more likely an abnormal frame. Figure 5 shows that the normality score obtained by MemAE immediately decreases when some anomalies appear in the video frame.

Due to the complexity of the video data, many general anomaly detection methods [28, 18, 48] without specific design cannot work well on videos. To show the effectiveness of the proposed memory module, we compare the proposed MemAE with many well-designed reconstruction based state-of-the-art methods including AE methods with 2D [11] and 3D convolution [45] (AE-Conv2D and AE-Conv3D), a temporally-coherent sparse coding method (TST) [26], a stacked recurrent neural network (StackRNN) [26] and many video anomaly detection baselines. The variants of MemAE are also compared as baselines.

Table 2 shows the AUC values on video datasets. MemAE produces much better results than TSC and StackRNN [26], which also apply the sparse regularization. The comparisons with AE and MemAE-nonSpar show that the memory module with the sparse addressing is steadily beneficial. Figure 6 visualizes the reconstruction error on one abnormal frame in UCSD-Ped2. The error map of MemAE significantly highlights the abnormal event (*i.e.* vehicle and bicycle moving on the sidewalk), inducing low normality score. However, AE reconstructs the anomaly well and induces some random errors.

The proposed MemAE obtains better or comparative performance than other methods, while our model solves a more general problem and can be flexibly applied to different types of data. By merely using the reconstruction error, the proposed method can obtain superior results with the minimum knowledge of the specific application. Even comparing with the method [24] (*i.e.* Frame-Pred in Table 2) that uses many non-reconstruction techniques specifically for video data, *e.g.* optical flow, frame prediction, and adversarial loss, the performance of the proposed MemAE is still comparable. Note that the purpose of our experiment is not pursuing the highest accuracy on certain applications but to demonstrate the advantages of the proposed improvement of AE, *i.e.* MemAE, for the general anomaly detection problem. Our study is orthogonal to that in [24] and can be readily incorporated into their system to boost the performance further. On the other hand, the techniques in [24] can also be used in the proposed MemAE.

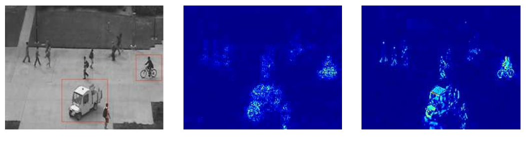


Figure 6. Reconstruction error of AE and MemAE on an abnormal frame of UCSD-Ped2. MemAE can significantly highlight the abnormal parts (in red bounding box) in the scene.

Robustness to the memory size We use the UCSD-Ped2 to study the robustness of the proposed MemAE to the memory size N. We conduct the experiments by using different memory size settings and show the AUC values in Figure 7. Given a large enough memory size, the MemAE can robustly produce plausible results.

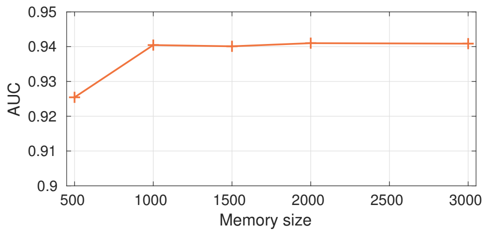


Figure 7. Robustness to the setting of memory size. AUC values of MemAE with different memory size on UCSD-Ped2 are shown.

Running time We empirically study the computing complexity of the proposed method on the video dataset UCSD-Ped2 with an NVIDIA GeForce 1080 Ti graphics card. The proposed MemAE averagely takes 0.0262 seconds for video anomaly detection of one frame (*i.e.* 38 fps), which is on par or faster than previous state-of-the-art deep learning based methods such as [24] using 0.04s, [26] using 0.02s and [37] using 0.05s[1]. Moreover, comparing to our baseline AE model that takes 0.0266s for each frame, our memory module (in MemAE) induces little additional computation time (*i.e.* 4×10^{-4}s per frame).

4.3. Experiments on Cybersecurity Data

To further validate the generalization of the proposed method, we experiment on a widely used cybersecurity dataset beyond the computer vision applications, *i.e.* KDDCUP99 10 percent dataset from the UCI repository [23].

[1]Running time of the compared methods are quoted from [24] for reference, which is produced using a faster graphics card than ours.

Table 3. Experimental results of different methods on the cybersecurity dataset KDDCUP.

Method\Metric	Precision	Recall	F_1
OC-SVM [35]	0.7457	0.8523	0.7954
DCN [42]	0.7696	0.7829	0.7762
DSEBM [43]	0.8619	0.6446	0.7399
DAGMM [48]	0.9297	0.9442	0.9369
AE	0.9328	0.9356	0.9342
MemAE-nonSpar	0.9341	0.9368	0.9355
MemAE	**0.9627**	**0.9655**	**0.9641**

Following the settings in [48], 80% of the samples labeled as "attack" in the original dataset are treated as normal samples. Each sample can be organized as a vector with 120 dimensions [48]. We use fully-connected layers (noted as FC) to implement the encoder and decoder as FC(120, 60)-FC(60, 30)-FC(30, 10)-FC(10, 3) and FC(3, 10)-FC(10, 30)-FC(30, 60)-FC(60, 120), in which FC(i, o) denotes the FC layer with input and output size i and o. Expect the last one, each FC layer is followed by a Tanh activation. The structure shares a similar capacity to the model in [48]. We set $N = 50$ and thus have a memory with the size of 50×3.

As suggested in [43, 48], we randomly sample 50% of data for training and the rest for testing. Only data samples from normal class are used for training. We compare the proposed method with previous state-of-the-art methods on the KDDCUP dataset, including OC-SVM [35], a deep clustering network (DCN) [42], DSEBM [43], DAGMM [48] and the baseline variants of MemAE. Following the standard protocol [48], the methods are evaluated using the average precision, recall and F_1 score after 20 runs. DAGMM and the proposed models perform very well because of the more effective data modeling. The proposed method obtains the superior performance since it can explicitly memorize the behavior patterns of "attack" samples.

4.4. Ablation Studies

In previous sections, extensive comparisons among MemAE and its variants, *i.e.* AE and MemAE-nonSpar, have proved the importance of the major components of the proposed method. In this section, we will conduct several further ablation studies to investigate other different components in details.

4.4.1 Study of the Sparsity-inducing Components

As introduced above, we use two components to induce sparsity of the memory addressing weights, *i.e.* the hard-thresholding shrinkage defined in Eq. (6) and the entropy loss $E(\cdot)$ in Eq. (10). We experiment to study the importance of each component by removing the other one. Table 4 records the AUC on the dataset UCSD-Ped2. As shown in Table 4, removing either the shrinkage operator or the entropy loss will degenerate the performance. Without the hard shrinkage, the model cannot directly encourage sparsity in testing, which may lead to non-sparse memory addressing weights with too much noise. Entropy loss plays a vital role when the under-trained model generates unoptimized addressing weights at the early stage of training.

Table 4. Ablation studies based on UCSD-Ped2 dataset.

Method	AUC
AE	0.9170
AE-ℓ_1	0.9286
MemAE-nonSpar	0.9293
MemAE w/o Shrinkage	0.9324
MemAE w/o Entropy loss	0.9372
MemAE	**0.9410**

4.4.2 Comparison with AE with Sparse Regularization

The sparse memory addressing in MemAE derives a flavor of the autoencoders that induce sparsity of the encoder output (activations). We thus conduct a straightforward experiment to compare MemAE with an autoencoder with sparse regularization on the encoded features, which is directly implemented by minimizing the ℓ_1-norm of the latent compressed feature, *i.e.* $\|\mathbf{z}\|_1$, during training, referred to as AE-ℓ_1, which shares the same encoder and decoder with MemAE. As shown in Table 4, the performance of AE-ℓ_1 is close to MemAE-nonSpar and superior to AE due to the sparsity-inducing regularization. However, AE-ℓ_1 still lacks a clear mechanism to encourage large reconstruction errors on anomalies or a powerful module to model the prototypical patterns of the normal samples, lead to worse performance than MemAE and other MemAE variants.

5. Conclusion

In this paper, we proposed a memory-augmented autoencoder (MemAE) to improve the performance of the autoencoder based unsupervised anomaly detection methods. Given an input, the propose MemAE first uses the encoder to obtain an encoded representation and then use the encoding as a query to retrieve the most relevant patterns in the memory for reconstruction. Since the memory is trained to record the prototypical normal patterns, the proposed MemAE can well reconstruct the normal samples and enlarge the reconstruction error of the anomalies, which strengths the reconstruction error as the anomaly detection criterion. Experiments on various datasets from different applications prove the generalization and effectiveness of the proposed method. In the future, we will investigate to use the addressing weight for anomaly detection. Considering that the proposed memory module is general and agnostic to the structures of the encoder and decoder, we will integrate it into more complicated base models and apply it on more challenging applications.

Acknowledgments This work was partially supported by iCetana Pty Ltd.

References

[1] Davide Abati, Angelo Porrello, Simone Calderara, and Rita Cucchiara. AND: Autoregressive novelty detectors. *arXiv preprint arXiv:1807.01653*, 2018.

[2] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, pages 153–160, 2007.

[3] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly detection using one-class neural networks. *arXiv preprint arXiv:1802.06360*, 2018.

[4] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[5] Yunqiang Chen, Xiang Sean Zhou, and Thomas S Huang. One-class svm for learning in image retrieval. In *The IEEE International Conference on Image Processing*, volume 1, pages 34–37. IEEE, 2001.

[6] Yong Shean Chong and Yong Haur Tay. Abnormal event detection in videos using spatiotemporal autoencoder. In *International Symposium on Neural Networks*, pages 189–196. Springer, 2017.

[7] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. In *Advances in Neural Information Processing Systems*, pages 9758–9769, 2018.

[8] Dong Gong, Mingkui Tan, Qinfeng Shi, Anton van den Hengel, and Yanning Zhang. MPTV: Matching pursuit-based total variation minimization for image deconvolution. *IEEE Transactions on Image Processing*, 28(4):1851–1865, 2019.

[9] Dong Gong, Mingkui Tan, Yanning Zhang, Anton Van den Hengel, and Qinfeng Shi. Blind image deconvolution by automatic gradient activation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1827–1836, 2016.

[10] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[11] Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K Roy-Chowdhury, and Larry S Davis. Learning temporal regularity in video sequences. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 733–742, 2016.

[12] Ryota Hinami, Tao Mei, and Shin'ichi Satoh. Joint detection and recounting of abnormal events by learning deep generic knowledge. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 3639–3647, 2017.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[14] Ian Jolliffe. Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer, 2011.

[15] Jaechul Kim and Kristen Grauman. Observe locally, infer globally: a space-time mrf for detecting abnormal activities with incremental updates. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[16] Youngjin Kim, Minjung Kim, and Gunhee Kim. Memorization precedes generation: Learning unsupervised gans with memory networks. *International Conference on Learning Representations (ICLR)*, 2018.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014.

[19] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[20] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[21] Roberto Leyva, Victor Sanchez, and Chang-Tsun Li. The lv dataset: A realistic surveillance video dataset for abnormal event detection. In *International Workshop on Biometrics and Forensics (IWBF)*, pages 1–6. IEEE, 2017.

[22] Chongxuan Li, Jun Zhu, and Bo Zhang. Learning to generate with memory. In *International Conference on Machine Learning (ICML)*, pages 1177–1186, 2016.

[23] Moshe Lichman et al. Uci machine learning repository, 2013.

[24] Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. Future frame prediction for anomaly detection–a new baseline. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[25] Cewu Lu, Jianping Shi, and Jiaya Jia. Abnormal event detection at 150 fps in matlab. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 2720–2727, 2013.

[26] Weixin Luo, Wen Liu, and Shenghua Gao. A revisit of sparse coding based anomaly detection in stacked rnn framework. *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[27] Vijay Mahadevan, Weixin Li, Viral Bhalodia, and Nuno Vasconcelos. Anomaly detection in crowded scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[28] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[30] Jack Rae, Jonathan J Hunt, Ivo Danihelka, Timothy Harley, Andrew W Senior, Gregory Wayne, Alex Graves, and Tim Lillicrap. Scaling memory-augmented neural networks with sparse reads and writes. In *Advances in Neural Information Processing Systems*, pages 3621–3629, 2016.

[31] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International Conference on Machine Learning (ICML)*, pages 4393–4402, 2018.

[32] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. Adversarially learned one-class classifier for novelty detection. In *The IEEE Conference*

on Computer Vision and Pattern Recognition, pages 3379–3388, 2018.

[33] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with memory-augmented neural networks. *International Conference on Machine Learning (ICML)*, 2016.

[34] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[35] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems*, pages 582–588, 2000.

[36] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[37] Radu Tudor Ionescu, Sorina Smeureanu, Bogdan Alexe, and Marius Popescu. Unmasking the abnormal events in video. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[38] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

[39] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *International Conference on Learning Representations (ICLR)*, 2015.

[40] Liang Xiong, Barnabás Póczos, and Jeff G Schneider. Group anomaly detection using flexible genre models. In *Advances in Neural Information Processing Systems*, pages 1071–1079, 2011.

[41] Dan Xu, Elisa Ricci, Yan Yan, Jingkuan Song, and Nicu Sebe. Learning deep representations of appearance and motion for anomalous event detection. *arXiv preprint arXiv:1510.01553*, 2015.

[42] Xi Yang, Kaizhu Huang, John Yannis Goulermas, and Rui Zhang. Joint learning of unsupervised dimensionality reduction and gaussian mixture model. *Neural Processing Letters*, 45(3):791–806, 2017.

[43] Shuangfei Zhai, Yu Cheng, Weining Lu, and Zhongfei Zhang. Deep structured energy based models for anomaly detection. In *International Conference on Machine Learning (ICML)*, pages 1100–1109, 2016.

[44] Bin Zhao, Li Fei-Fei, and Eric P Xing. Online detection of unusual events in videos via dynamic sparse coding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3313–3320, 2011.

[45] Yiru Zhao, Bing Deng, Chen Shen, Yao Liu, Hongtao Lu, and Xian-Sheng Hua. Spatio-temporal autoencoder for video anomaly detection. In *ACM on Multimedia Conference*, pages 1933–1941. ACM, 2017.

[46] Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *ACM SIGKDD*, pages 665–674. ACM, 2017.

[47] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(5):363–387, 2012.

[48] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.