AMAD: Adversarial Multiscale Anomaly Detection on High-Dimensional and Time-Evolving Categorical Data

Zheng Gao
Indiana University Bloomington
gao27@indiana.edu

Lin Guo
Alibaba Group
lin.gl@alibaba-inc.com

Chi Ma
Alibaba Group
beiji.mc@alibaba-inc.com

Xiao Ma
Alibaba Group
maxiao.mx@alibaba-inc.com

Kai Sun
Alibaba Group
luchen.sk@alibaba-inc.com

Hang Xiang
Alibaba Group
xingzhi.xh@alibaba-inc.com

Xiaoqiang Zhu
Alibaba Group
xiaoqiang.zxq@alibaba-inc.com

Hongsong Li
Alibaba Group
hongsong.lhs@alibaba-inc.com

Xiaozhong Liu
Indiana University Bloomington
liu237@indiana.edu

ABSTRACT

Anomaly detection is facing with emerging challenges in many important industry domains, such as cyber security and online recommendation and advertising. The recent trend in these areas calls for anomaly detection on time-evolving data with high-dimensional categorical features without labeled samples. Also, there is an increasing demand for identifying and monitoring irregular patterns at multiple resolutions. In this work, we propose a unified end-to-end approach to solve these challenges by combining the advantages of Adversarial Autoencoder and Recurrent Neural Network. The model learns data representations cross different scales with attention mechanisms, on which an enhanced two-resolution anomaly detector is developed for both instances and data blocks. Extensive experiments are performed over three types of datasets to demonstrate the efficacy of our method and its superiority over the state-of-art approaches.

KEYWORDS

Anomaly Detection, Adversarial Autoencoder, High-dimensional Data

ACM Reference format:

Zheng Gao, Lin Guo, Chi Ma, Xiao Ma, Kai Sun, Hang Xiang, Xiaoqiang Zhu, Hongsong Li, and Xiaozhong Liu. 2019. AMAD: Adversarial Multiscale Anomaly Detection on High-Dimensional and Time-Evolving Categorical Data. In *Proceedings of 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data, Anchorage, AK, USA, August 5, 2019 (DLP-KDD'19),* 8 pages.

1 INTRODUCTION

Anomaly detection aims at identifying outliers or irregular patterns which are inconsistent with the majority of data. It can provide a wide range of applications, from capturing rare events or unusual observations to protecting a complex system against failures or attacks. Recent trend in many important industrial domains, such as online recommendation and advertising (as illustrated in Figure 1), online financial service and cyber security, has set four unprecedented challenges for anomaly detection. ***First***, as data is changed with time, there is no gold standard for anomalous data across all time periods. ***Second***, labeled anomalous samples are rarely available. ***Third***, data format can be very complex, for example, a compound of attributes consisting of categorical ids with extremely high dimension. The sparse, sophisticated and noisy couplings among massive features make it very difficult to recognize the underlying patterns through handcrafted rules or feature engineering. ***Fourth***, a systematic monitoring may require detecting anomalous events at different resolutions for different needs. The data patterns can vary with scales. Although straightforward, aggregating small-scale detection results for larger-scale detection is not guaranteed to be effective. For example, detecting collective patterns such as phase distortion.

Deep learning has drawn immense attention in the field of anomaly detection. Deep neural network has the potential to automatically learn complex feature representations, thus making it possible to train an anomaly detector in an end-to-end fashion with less nontrivial expertise feature engineering. Inspired by Generative Adversarial Network (GAN) [13] and Adversarial Autoencoder [20], some researches have been reported to adversarially train a pair of neural networks (generator and discriminator) through unsupervised or semi-supervised learning to construct an anomaly detector [1, 23, 31]. The networks' losses, especially adversarial loss and reconstruction loss, are used to identify anomalies. By design, normal samples should follow the distribution close to the majority of the training data, and thus obtain lower losses than anomalous samples. Assuming that the training data is normal, this type of approaches do not require labeled anomalies for training. They are able to solve the second challenge (lack of label) listed above but do not cover the other three.

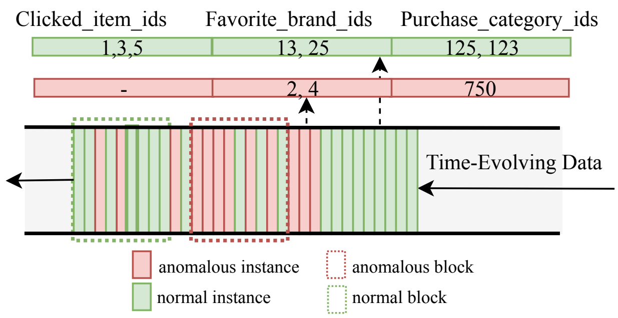

Figure 1: An example to show the data format in an online recommendation system. Each data instance contains multiple attributes composed of a group of categorical features. Either system failure or fraud attack can generate anomalous data with irregular values.

In this paper, following the emerging idea of adversarially learned anomaly detection, we present an adversarial multiscale anomaly detector (AMAD) to tackle the aforementioned challenges in an end-to-end manner. We train a pair of deep encoder-decoder generator and discriminator to fit the normal patterns of the unlabeled training data (*Challenge 2*), and using a compound loss as anomaly score for inference. We combine sequential and hierarchical representation learning to detect anomalies at two different scales (*Challenge 4*) for time-evolving high-dimensional categorical data (*Challenge 1,3*). The main contributions of this work are highlighted as follows [1]:

- To the best of our knowledge, AMAD is the first unified end-to-end approach to tackle the aforementioned important challenges. Especially, our work is the first attempt to extend adversarial anomaly detector to the scenario of complex high-dimensional categorical data.
- We introduce a multiscale data representation learning mechanism. Patterns are extracted and inspected cross a range of scales, from single features of an individual instance up to data blocks. This produces an enhanced two-resolution anomaly detector for both individual instances and data blocks.
- We report extensive experiments on three types of datasets, validating that our model outperforms the state-of-arts notably. Moreover, we conduct ablation studies to prove the efficacy of the key components in our model.

2 RELATED WORKS

Deep learning has been widely applied in all research topics such as ranking [12], graph mining [11] and text generation [30], etc. As a fundamental one, anomaly detection has been extensively studied via unsupervised or semi-supervised deep approaches. iForest [18], one of the most famous approaches, utilizes a tree-based structure to split data randomly and ranks data points as anomalous based on how easy they get isolated. Affiliated with Support Vector Machine (SVM) family, one-class SVM classifiers [5, 27, 28] use designed kernels to project data to a latent space and search for a best hyperplane to set anomalies apart. Derived from these works, kernel-based one-class classification is further combined with deep neural network [4, 22] to automatically extract useful features from massive complex data.

Deep learning attracts increasing attentions for the past decade. As a basic type of deep learning framework, autoencoder has already widely applied on anomaly detection [2, 3, 25, 32]. It learns to compress the input data with multiple hidden layers and reconstruct the input data through an encoding-decoding mechanism. Trained solely on normal data, autoencoder fails to reconstruct anomalous sample and produces large reconstruction error that can be used to identify anomaly. Furthermore, an autoencoder ensemble with adaptive sampling is proposed to improve the robustness on noisy data [6].

Recently, Generative Adversarial Networks rise up as a popular track in deep learning [15, 23, 26]. Typically, a GAN-based model consists of two parts, i.e., generator and discriminator. The generator learns a representation to resemble the original input data, while the discriminator is trained to distinguish between the resembled and original inputs. The adversarial training enhances the model's ability of learning the distribution of input normal data, and is proven to be very effective for identifying anomalous or novel data.

Combining GAN and autoencoder, Adversarial Autoencoder [20] offers an alternative way for unsupervised or semi-supervised anomaly detection. Unlike GAN approaches which learns a distribution to generate discrete samples, Adversarial Autoencoder uses autoencoder as the generator to learn to resemble data. By mapping the input to latent space and remapping back to input data space (reconstruction), it enables not only better reconstruction but also control over latent space [8, 21]. Taking this track of thoughts, BiGAN [9] and ALI [10] both apply variational autoencoder as the generators in their models to optimize the distribution of normal data. Two following works [1, 31] combine both GAN and Adversarial Autoencoder components to jointly train an anomaly detector and use the reconstruction errors as the criteria to judge whether testing data is anomalous or not.

3 METHOD

3.1 Overview

The framework of our model is sketched in Figure 2. Our approach adversarially trains an anomaly detection model on unlabeled data, with the assumption that the training data is normal (at least mostly normal). The input data for the model is of a hierarchical four-level structure (also illustrated in Figure 2). We use the model to detect anomalies at the top two levels (instance and block). Since the model is trained to fit the distribution of normal data, the anomalies should have higher loss than normal data. Therefore, we use the loss to infer anomalies [1, 15, 23, 26, 31].

In the following sections, we first introduce the multiscale representation learning across different levels. Second, we describe the adversarial learning architecture. In the end, we explain how we train the model and use the model for inference.

3.2 Multiscale Representation Learning

Our model hierarchically learns representations for the structured input data, from feature, attribute, instance, up to instance block level. We implement attention mechanism [17, 29, 33] to summarize informations with distributed weights of importance to form the

[1] Our codes and datasets will be made available at publication time.

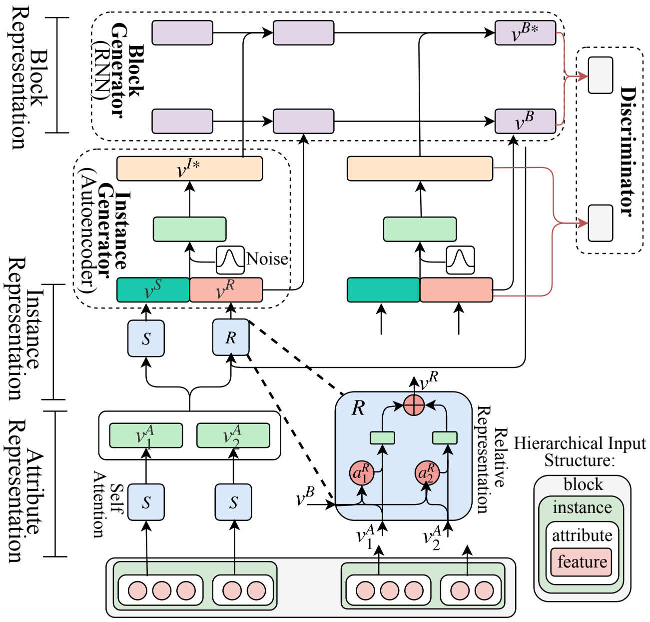

Figure 2: The overall architecture of AMAD

next-level representation, so that most important informations are extracted to the high level.

3.2.1 Feature and Attribute Representation. For the input layer, sparse embedding is implemented to embed each categorical feature to a fixed-size dense vector v^F, which is automatically learned during the training process. For each attribute, its representation vector v^A is extracted from all the embedding vectors of its input feature collection $\{v_1^F, ..., v_{N^A}^F\}$ with a self-attention mechanism [17]:

$$\begin{aligned} e_i^F &= (u^F)^\intercal \tanh(\mathbf{W}^F v_i^F + b^F), \\ a_i^F &= \frac{\exp(e_i^F)}{\sum_{j=1}^{N^A} \exp(e_j^F)}, \\ v^A &= \sum_{i=1}^{N^A} a_i^F v_i^F, \end{aligned} \tag{1}$$

where $\mathbf{W}^F$, b^F and u^F are trainable weight matrix, bias vector and attention vector, respectively. N^A denotes the number of the input features belonging to the attribute. v_i^F, e_i^F and a_i^F denote the embedding vector, attention score, and normalized attention score of the ith feature, respectively.

3.2.2 Instance Representation. Based on the attribute vectors, we construct the higher-level representation for each input instance from two channels, i.e., self representation and relative representation against the previous data block.

The self representation vector, v^S, is extracted from the instance's attribute representations $\{v_1^A, ..., v_{N^I}^A\}$:

$$\begin{aligned} e_i^A &= (u^A)^\intercal \tanh(\mathbf{W}^A v_i^A + b^A), \\ a_i^A &= \frac{\exp(e_i^A)}{\sum_{j=1}^{N^I} \exp(e_j^A)}, \\ v^S &= \sum_{i=1}^{N^I} a_i^A v_i^A, \end{aligned} \tag{2}$$

where $\mathbf{W}^A$, b^A and u^A are trainable weight matrix, bias vector and attention vector, respectively. N^I denotes the number of attributes. v_i^A, e_i^A and a_i^A denote the embedding vector, attention score, and normalized attention score of the ith attribute, respectively.

The relative representation vector, v^R, is calculated by comparing the instance's attributes with the previous block vector. It is designed to enable the model to extract instance's relative patterns against the larger-scale collective patterns of the data:

$$\begin{aligned} e_i^R &= (u^R)^\intercal \tanh(\mathbf{W}^R [f(v_i^A), v^{Mem}] + b^R), \\ a_i^R &= \frac{\exp(e_i^R)}{\sum_{j=1}^{N^I} \exp(e_j^R)}, \\ v^R &= \sum_{i=1}^{N^I} a_i^R v_i^A, \end{aligned} \tag{3}$$

where the square brackets [,] denotes the concatenation operation. The transformation function $f(\cdot)$ is the Leaky ReLU activation function. $\mathbf{W}^R$, b^R and u^R are trainable weight matrix, bias vector and attention vector, respectively. e_i^A and a_i^A are the attention score and normalized attention score of the ith attribute, respectively. v^{Mem} is the memory vector from the previous data block and its calculation will be described in the next section.

For the output of this module, we concatenate the two latent vectors to form the final representation vector of the instance:

$$v^I = \text{batch\_norm}([v^S, v^R]). \tag{4}$$

3.2.3 Block Representation. Furthermore, we go beyond instance level and implement a Recurrent Neural Network (RNN) cell to capture the long-term collective patterns of the sequential instances. For each data block, the representation can be calculated as:

$$v_i^B = \text{RNN}(f(v_i^I), v_{i-1}^B), i = 1, ..., N^B, \tag{5}$$

where N^B is the instance number in the block.

The last hidden state, denoted by v^B, contains the latest and most information about the data's collective patterns over time. For this reason, we use v^B as a representation for the block to improve the block-level anomaly detection (will be revisited in the following paragraphs). Moreover, it is also used as both the memory vector v^{Mem} (in Eq. 3) and the initial hidden state for the next block.

3.3 Adversarial Learning

Following the idea of Adversarial Autoencoder, we build an adversarial learning architecture to learn the intrinsic patterns of the training data for both instance and block levels. On one hand, the encoder-decoder generator part learns to generate resembled representations of the inputs. In this way, cycle consistency [31] is enforced in the latent space. On the other hand, the discriminator tries to distinguish the real and resembled representations.

3.3.1 Instance Generator. We use an autoencoder to generate resembled instance vectors. The autoencoder first encodes the instance representation vector v^I into a hidden space, and subsequently decodes it back to reconstruct a representation vector v^{I*}:

$$\begin{aligned} h^{enc} &= \mathbf{W}^{enc}(f(v^I) + \Delta) + b^{enc}, \\ v^{I*} &= f(\mathbf{W}^{dec} f(h^{enc}) + b^{dec}) - \Delta, \end{aligned} \tag{6}$$

where $\mathbf{W}$s and bs are the trainable weights and biases, respectively.

The performance of autoencoder is vulnerable to the noise in training data [32]. In order to get a more robust model, we add a standard Multivariate Gaussian random noise $\Delta \sim \mathcal{N}_d(0, E)$ into the encode-decode process, where E is the identity matrix and d is the dimension of instance vector.

3.3.2 Block Generator. To introduce adversarial learning for long-term patterns, we also reconstruct a resembled vector per block for v^B:

$$v_i^{B*} = \text{RNN}(f(v_i^{I*}), v_{i-1}^{B*}), i = 1, ..., N^B. \tag{7}$$

To be consistent with the calculation of real block vector v^B, here, we don't train the weight and bias for the RNN cell, but directly copy the values of the corresponding parameters used for Eq. 5. Similarly, the last hidden state is taken as the final resembled vector of the current block, denoted by v^{B*}.

3.3.3 Discriminator. Following the standard setting of binary classification, we build two one-layer neural network classifiers for both instance and block levels:

$$\hat{y}^I = \sigma(\mathbf{W}^I x^I + b^I) \text{ with } x^I \in \{v^I, v^{I*}\} \tag{8}$$

and

$$\hat{y}^B = \sigma(\mathbf{W}^B x^B + b^B) \text{ with } x^B \in \{v^B, v^{B*}\}, \tag{9}$$

where $\mathbf{W}$s and bs are the trainable weights and biases, respectively. $\sigma(\cdot)$ denote sigmoid activation function.

3.4 Training and Inference

3.4.1 Training. In the training stage, we assume all training data are normal data to train our model in the unsupervised manner. As generative adversarial training is hard to converge, we don't minimize the generator loss and discriminator loss at the same time. Instead, we minimize the two losses in an alternative process: first holding the discriminator loss $\mathcal{L}_{\mathcal{D}}$ and minimizing generator loss $\mathcal{L}_{\mathcal{G}}$ for several steps, and then minimizing discriminator $\mathcal{L}_{\mathcal{D}}$ with the generator loss $\mathcal{L}_{\mathcal{G}}$ being held.

For the generator loss, we use the sigmoid cross entropy [7] between real and resembled vectors

$$\mathcal{L}_{\mathcal{G}}^I = \sigma(v^I)^\intercal log(\sigma(v^{I*})) + (\mathbb{1} - \sigma(v^I))^\intercal log(\mathbb{1} - \sigma(v^{I*})) \tag{10}$$

as the instance generator loss, and

$$\mathcal{L}_{\mathcal{G}}^B = \sigma(v^B)^\intercal log(\sigma(v^{B*})) + (\mathbb{1} - \sigma(v^B))^\intercal log(\mathbb{1} - \sigma(v^{B*})) \tag{11}$$

as the block generator loss. The total generator loss to minimize is the sum of block generator loss and the average of its N^B instance generator losses:

$$\mathcal{L}_{\mathcal{G}} = \frac{1}{N^B} \sum_{i=1}^{N^B} \mathcal{L}_{\mathcal{G},i}^I + \mathcal{L}_{\mathcal{G}}^B. \tag{12}$$

For the discriminator loss, under standard setting of binary classification, we also use cross entropy based on the output of Equations 8 and 9

$$\mathcal{L}_{\mathcal{D}}^I = y^I log(\hat{y}^I) + (1 - y^I) log(1 - \hat{y}^I) \tag{13}$$

as the instance discriminator loss, and

$$\mathcal{L}_{\mathcal{D}}^B = y^B log(\hat{y}^B) + (1 - y^B) log(1 - \hat{y}^B) \tag{14}$$

as the block discriminator loss. $y \in \{y^I, y^B\}$ is defined as: $y = 1$ for real vectors and $y = 0$ for resemble vectors. In each optimization step, the total discriminator loss of a data block to minimize is:

$$\mathcal{L}_{\mathcal{D}} = \frac{1}{N^B} \sum_{i=1}^{N^B} \mathcal{L}_{\mathcal{D},i}^I + \mathcal{L}_{\mathcal{D}}^B. \tag{15}$$

3.4.2 Inference. For inference, we use compound loss as the output anomaly score to measure the degree of abnormality. Because the model lowers down the total loss by learning to fit normal data patterns during training. Abnormal data will produce higher loss since the model fails to fit abnormal patterns. The anomaly score of an instance is given by:

$$z^I = \mathcal{L}_{\mathcal{G}}^I + \beta \cdot \mathcal{L}_{\mathcal{D}}^I. \tag{16}$$

The anomaly score for a block is calculated by including both the block-level losses and the average instance anomaly score within the block:

$$z^B = \mathcal{L}_{\mathcal{G}}^B + \beta \cdot \mathcal{L}_{\mathcal{D}}^B + \gamma \cdot \frac{1}{N^B} \sum_{i=1}^{N^B} z_i^I. \tag{17}$$

Two weight parameters β and γ are introduced to balance the influences from the different terms.

4 EXPERIMENT SETUP

4.1 Datasets

Three datasets are utilized to illustrate the performance of the proposed method. Their statistics are shown in Table 1, and more details are described in Appendix.

Synthetic dataset: The Synthetic data is generated by adding random noises to multi-dimensional zigzag signals of discrete integers. The anomalies are constructed by either randomly generating numbers or randomly copying training instances.

Public dataset[2]: It is a public dataset about positions in the 'connect-4' game. We use the instances labelled with 'win' as normal data and the instances labelled with 'loss' as anomaly data. Note that there is no sequential relation within the data.

Industrial dataset[3]: The Industrial dataset is constructed by user behavior data from a real-world online recommendation system, which is very important for many tasks, such as click-through rate prediction [19, 33]. The instances are collected over 10 consecutive

[2] https://archive.ics.uci.edu/ml/datasets/Connect-4

[3] The Industrial dataset is published at https://tianchi.aliyun.com/dataset/dataDetail?dataId=27665.

days, and stored in the order of timestamp. As illustrated in Figure 1, each instance consists of multiple attributes about user's past behaviors, e.g., clicked items in the past 3 days, favorite brands in the past week, etc. Each attribute contains a group of categorical ids in representation of the corresponding items, brands, etc. It's impractical to get a dataset with enough well labeled anomalies from real-world production. Thus, we mimic anomalies by simulating real conditions (attributes are polluted by errors in upstream data pipeline). The anomalies are generated by deleting the records of a random selected attribute, or replacing the records with random ids.

Dataset	#Dimension	#Attribute	#Normal	#Anomaly
Synthetic	30	3	10,000	1,000
Public	192	64	44,000	4,000
Industrial	440,512	8	783,000	25,000

Table 1: Statistics of three datasets used in the paper. The dimension denotes the total number of all the distinct categorical feature IDs for the entire dataset.

Method	Reference	Category
OCSVM[4]	[27]	SVM
iForest[4]	[18]	Tree ensemble
RDA[5]	[32]	Autoencoder
OCNN[6]	[4]	Deep SVM
ALAD[7]	[31]	GAN
GANomaly[8]	[1]	GAN
ALOCC[9]	[23]	GAN

Table 2: Baseline methods in this paper.

For the Public dataset, the normal data are randomly split for either training or testing, while the testing normal data is mixed with the anomalous data to form the final testing set. For the Synthetic and Industrial datasets, the former part (the majority) of the normal data is used for training, whereas the last small portion is used separately and mixed with anomaly samples for testing. To better cover the high-dimension space outside the normal data and test the models' performance more efficiently, we include a large ratio of anomalies for the test data. All the testing datasets have a half-to-half ratio of normal and anomalous instances. Unbalanced test set can be made by down sampling anomalies, and the corresponding performance metrics such as recall and precision can be calculated by adjusting the reported results with sample ratio. For block-level detection, we define that a testing data block is anomalous if more than 50% instances in a block are anomalous. Otherwise the block is defined as normal.

[4]https://scikit-learn.org/
[5]https://github.com/zc8340311/RobustAutoencoder
[6]https://github.com/raghavchalapathy/oc-nn
[7]https://github.com/houssamzenati/Efficient-GAN-Anomaly-Detection
[8]https://github.com/samet-akcay/ganomaly
[9]https://github.com/khalooei/ALOCC-CVPR2018

4.2 Settings

4.2.1 Parameters and Metrics. We report the results with best hyper-parameters from grid search. We train our model using RMSProp optimizer with learning rate as 0.01. We set block size as 100, $\beta = 0.3$ and $\gamma = 0.05$ for calculating the compound anomaly score (Eqs. 16 and 17).

In this paper, we report Accuracy, F1-score and AUROC as the metrics to evaluate model performance. With the anomaly scores of all testing data, we firstly calculate AUROC score by considering all possible thresholds and subsequently pick the optimal threshold as defined in [14]. Based on the optimal threshold, Accuracy and F1-score are calculated in the end.

4.2.2 Baselines. As listed in Table 2, we selected 7 state-of-art methods for comparison with proposed model. These models only output anomaly score for each individual instance. We use the average score of the instances in a block as the block-level anomaly score. We also want to note that the baseline models are not originally designed for the complex high-dimensional categorical data such as our Industrial dataset. Therefore, for the Industrial dataset, we embed the instances into vectors ahead with Doc2Vec [16], and use the embedded vectors of instances as the input for the baseline methods.

5 RESULTS

In this section, we report the experimental results to demonstrate our approach's superiority over the other methods. Moreover, we present studies to verify the effects of the important characteristics of our model.

5.1 Performance of the Full Model

We run our model ten times and report the average evaluation results on instance-level anomaly detection and block-level anomaly detection in Tables 3 and 4. To verify our model's superiority, we calculate the performance differences between our model and the best baseline on each metric for all the runs, and apply a T-test to check whether the performance difference is significantly above 0 or not.

We can find that the GAN-based models perform best among the baselines, while our model outperforms all the baselines with respect to all the metrics and datasets. Moreover, our model displays larger advantages for block-level detection. Clearly, the block-level anomaly detection gets more benefits from our unified multiscale approach.

5.2 Random Noise in Autoencoder

The performance of autoencoder can be deteriorated due to the noises in the training data. We add a random noise into the autoencoder (Eq. 6), to make it more robust. To check the utility of the added noises, we retrain an ablated model by removing the Δ in Eq. 6. As shown by the results in the first rows (*–Noise*) of Tables 5 and 6, adding noise clearly improves the generalization performance for all the tests.

Model	Synthetic Dataset			Public Dataset			Industrial Dataset		
	Accuracy	F1-macro	AUROC	Accuracy	F1-macro	AUROC	Accuracy	F1-macro	AUROC
OCSVM	0.650	0.644	0.641	0.578	0.577	0.609	0.591	0.591	0.623
iForest	0.650	0.650	0.671	0.576	0.575	0.618	0.561	0.559	0.589
RDA	0.655	0.652	0.704	0.564	0.563	0.564	0.530	0.530	0.538
OCNN	0.550	0.549	0.533	0.578	0.577	0.577	0.624	0.584	0.624
ALAD	0.664	0.664	0.705	0.650	0.650	0.706	0.618	0.618	0.661
GANomaly	0.646	0.644	0.703	0.676	0.676	0.709	0.610	0.610	0.657
ALOCC	0.638	0.637	0.692	0.682	0.681	0.703	0.612	0.611	0.648
AMAD	**0.680***	**0.681***	**0.717***	**0.700***	**0.689***	**0.730***	**0.644***	**0.643***	**0.691***

Table 3: Results of instance anomaly detection on three datasets. Symbol '*' highlights the cases where our model significantly beats the best baseline with p value smaller than 0.01.

Model	Synthetic Dataset			Public Dataset			Industrial Dataset		
	Accuracy	F1-macro	AUROC	Accuracy	F1-macro	AUROC	Accuracy	F1-macro	AUROC
OCSVM	0.543	0.528	0.555	0.580	0.573	0.562	0.572	0.561	0.581
iForest	0.672	0.660	0.674	0.540	0.539	0.562	0.501	0.499	0.494
RDA	0.644	0.633	0.680	0.538	0.535	0.543	0.520	0.508	0.499
OCNN	0.512	0.508	0.498	0.572	0.554	0.561	0.626	0.577	0.648
ALAD	0.600	0.599	0.629	0.655	0.650	0.683	0.600	0.596	0.620
GANomaly	0.563	0.555	0.574	0.652	0.642	0.679	0.579	0.557	0.580
ALOCC	0.641	0.625	0.640	0.512	0.512	0.523	0.581	0.576	0.605
AMAD	**0.764***	**0.767***	**0.745***	**0.660***	**0.659***	**0.746***	**0.655***	**0.655***	**0.674***

Table 4: Results of block anomaly detection on three datasets. Symbol '*' highlights the cases where our model significantly beats the best baseline with p value smaller than 0.01.

Model	Synthetic Dataset			Public Dataset			Industrial Dataset		
	Accuracy	F1-macro	AUROC	Accuracy	F1-macro	AUROC	Accuracy	F1-macro	AUROC
– Noise	-4.0	-4.2	-0.1	-7.5	-8.7	-2.8	-2.4	-2.2	-3.2
– RelRep	-5.9	-6.3	-5.9	-1.7	-1.3	-0.4	-4.4	-6.9	-8.5

Table 5: Instance-level performance differences between the ablated model and the full model. Results are scaled by a factor of 100.

Model	Synthetic Dataset			Public Dataset			Industrial Dataset		
	Accuracy	F1-macro	AUROC	Accuracy	F1-macro	AUROC	Accuracy	F1-macro	AUROC
– Noise	-6.4	-7.0	-0.3	-3.0	-5.7	-0.7	-2.2	-2.2	-0.6
– RelRep	-14.2	-14.8	-2.6	0.0	-1.1	-0.6	-3.9	-7.0	-1.9
– BlockLoss	-10.0	-11.1	-7.4	-2.7	-2.1	-0.9	-6.2	-5.3	-4.4

Table 6: Block-level performance differences between the ablated model and the full model. Results are scaled by a factor of 100.

5.3 Relative Representation of Instance

We introduce the relative representation v^R (Eqs. 3 and 4) to improve the pattern recognition. To justify this, we retrain an ablated model (*–RelRep*) without this module, i.e., deleting Eq. 3 and removing v^R from Eq. 4. It leads a big drop in the performance, as shown in the second rows of Tables 5 and 6. Through comparing the instance with block representation, v^R notably enriches the information extracted at high level.

5.4 Block Loss for Block-Level Detection

To testify the effect of the block loss for detecting anomalous blocks. In consistent with baseline approaches, we remove the first two terms in Eq. 17, and only use the average instance anomaly score. As listed in the third row of Table 6 (*–BlockLoss*), the performance drops drastically down to the level very close to the baselines! The block loss adds detection of the collective patterns to our model, which is of great importance for detecting block-level anomaly.

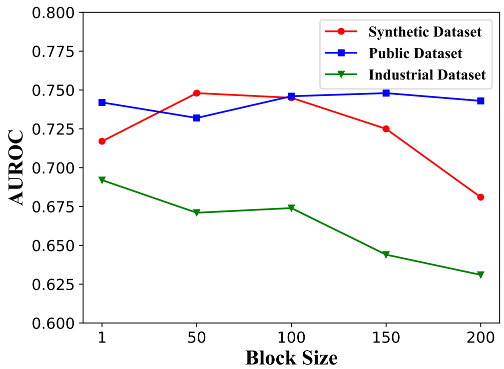

Figure 3: The full model's performance of block level detection as a function of testing block size on the three datasets.

5.5 Performance on Non-Sequential Data

Different with the two time-evolving datasets (Synthetic and Industrial dataset), the Public dataset is used to test the models under a simpler scenario, i.e., categorical data without sequential patterns. As expected, on this non-sequential data, the sequence-related components of our model has very weak impact on the performance (–*RelRep* and –*BlockLoss* in Tables 5 and 6). Even though, our model still outperforms all the baselines (Tables 5 and 6), which demonstrates the intrinsic superiority of the hierarchical representation learning structure of our model.

5.6 Block Size for Detection

We also tested how the testing block size affects our model's detection performance (as shown in Figure 3). For the Public dataset, there is no significant correlation between block size and the AUROC score, in consistent with the very weak effect of sequence-related components (as discussed in the last subsection). For both Synthetic and Industrial datasets, we can see the performance drops for block size > 100, where the testing block size exceeds the training block size. Also, for the left most case where block size = 1, the AUROC scores are very close to the corresponding results in Table 3. Including block losses (Eq. 17) barely affect detection for individual instances.

6 SUMMARY AND DISCUSSION

In this paper, we present AMAD, a multiscale Adversarial Autoencoder for anomaly detection at different levels on high-dimensional and time-evolving categorical data. We demonstrate the effectiveness of our method by extensive experiments on datasets of different sizes and scenarios.

Real-world streaming data can have severe non-stationary data drift problem, which may require to keep the model updated in time through online incremental learning [24]. In this work, we have trained our model in the setting of online incremental learning, i.e., processing the time-ordered data block by block. For future work, a more comprehensive study on online incremental learning will be followed. Also, we will work on extending the approach to more complex industrial scenarios, i.e., multiscale detection cross large span of resolutions (from instance and block level to hour and day levels), large-scale distributed data process.

REFERENCES

[1] Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. 2018. GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training. *arXiv preprint arXiv:1805.06725* (2018).
[2] Jerone TA Andrews, Edward J Morton, and Lewis D Griffin. 2016. Detecting anomalous data using auto-encoders. *International Journal of Machine Learning and Computing* 6, 1 (2016), 21.
[3] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. 2017. Robust, deep and inductive anomaly detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 36–51.
[4] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. 2018. Anomaly Detection using One-Class Neural Networks. *arXiv preprint arXiv:1802.06360* (2018).
[5] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* 2, 3 (2011), 27.
[6] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga. 2017. Outlier detection with autoencoder ensembles. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 90–98.
[7] Antonia Creswell, Kai Arulkumaran, and Anil A Bharath. 2017. On denoising autoencoders trained to minimise binary cross-entropy. *arXiv preprint arXiv:1708.08487* (2017).
[8] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. 2018. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine* 35, 1 (2018), 53–65.
[9] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. 2016. Adversarial feature learning. *arXiv preprint arXiv:1605.09782* (2016).
[10] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. 2016. Adversarially learned inference. *arXiv preprint arXiv:1606.00704* (2016).
[11] Zheng Gao, Gang Fu, Chunping Ouyang, Satoshi Tsutsui, Xiaozhong Liu, and Ying Ding. 2018. edge2vec: Learning Node Representation Using Edge Semantics. *arXiv preprint arXiv:1809.02269* (2018).
[12] Zizhe Gao, Zheng Gao, Heng Huang, Zhuoren Jiang, and Yuliang Yan. 2018. An End-to-end Model of Predicting Diverse Ranking On Heterogeneous Feeds. (2018).
[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.
[14] Farrokh Habibzadeh, Parham Habibzadeh, and Mahboobeh Yadollahie. 2016. On determining the most appropriate test cut-off value: the case of tests with continuous results. *Biochemia medica: Biochemia medica* 26, 3 (2016), 297–307.
[15] Mark Kliger and Shachar Fleishman. 2018. Novelty Detection with GAN. *arXiv preprint arXiv:1802.10560* (2018).
[16] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*. 1188–1196.
[17] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* (2017).
[18] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2012. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6, 1 (2012), 3.
[19] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 1137–1140.
[20] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644* (2015).
[21] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).
[22] Lukas Ruff, Nico Görnitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Robert Vandermeulen, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep one-class classification. In *International Conference on Machine Learning*. 4390–4399.
[23] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. 2018. Adversarially Learned One-Class Classifier for Novelty Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3379–3388.
[24] Doyen Sahoo, Quang Pham, Jing Lu, and Steven CH Hoi. 2017. Online deep learning: Learning deep neural networks on the fly. *arXiv preprint arXiv:1711.03705* (2017).
[25] Mayu Sakurada and Takehisa Yairi. 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd*

Workshop on Machine Learning for Sensory Data Analysis. ACM, 4.
[26] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. 2017. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*. Springer, 146–157.
[27] Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. 2001. Estimating the Support of a High-Dimensional Distribution. *Neural Comput.* 13, 7 (July 2001), 1443–1471. DOI: http://dx.doi.org/10.1162/089976601750264965
[28] David MJ Tax and Robert PW Duin. 2004. Support vector data description. *Machine learning* 54, 1 (2004), 45–66.
[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.
[30] Yongzhen Wang, Xiaozhong Liu, and Zheng Gao. 2019. Neural Related Work Summarization with a Joint Context-driven Attention Mechanism. *arXiv preprint arXiv:1901.09492* (2019).
[31] Houssam Zenati, Manon Romain, Chuan-Sheng Foo, Bruno Lecouat, and Vijay Chandrasekhar. 2018. Adversarially Learned Anomaly Detection. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 727–736.
[32] Chong Zhou and Randy C Paffenroth. 2017. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 665–674.
[33] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2018. Deep Interest Evolution Network for Click-Through Rate Prediction. *arXiv preprint arXiv:1809.03672* (2018).

7 APPENDIX

7.1 Details of Datasets

Synthetic dataset: We initialize the first instance with three categorical ids '0,10,20', and then generate the following instances by add 1 on each ID of the previous instance. If the ID surpasses 30, it will be subtracted by 30 and substituted by the remainder. Noises are introduced to the deterministic signals, by randomly selecting 10% IDs and adding random noises $\in \{-1, 1\}$ onto their original values. This process is repeated 220 times with a period of 50, generating 11000 normal instances. We use the first 9000 normal instances as the training data and the remaining 2000 for testing. In the testing set, we randomly select 1000 instance and replace them with anomalies. The anomalies are constructed by either randomly generating IDs or copying randomly selected training instances.

Public dataset: It is about positions in the 'connect-4' game. Each feature in an instance refers to one of three choices (taken, not taken, blank) on a position. Each instance refers to a possible choice permutation. We use the instances labelled with 'win' as normal data and the instances labelled with 'loss' as anomaly data. We randomly select 40000 normal instance for training, while randomly mix the other 4000 with the 4000 anomalies to form the testing set.

Industrial dataset: The Industrial dataset is constructed by user behavior data from our online recommendation system. The user behavior record is updated according to user's latest behaviors. Whenever the system receives an impression request from a user, a instance is generated. The data is collected over 10 consecutive days, and stored in the order of timestamp. All the real-world data is assumed to be normal. We use the first 758000 normal instances as the training data and the remaining 50000 for testing. In the testing set, we randomly select 25000 instances and replace them with anomalies. The anomalies are generated by deleting the records under a random selected attribute, replacing the records with random IDs.