Deep into Hypersphere: Robust and Unsupervised Anomaly Discovery in Dynamic Networks

Xian Teng[1], **Muheng Yan**[1], **Ali Mert Ertugrul**[1,2], **Yu-Ru Lin**[1*]
[1] School of Computing and Information, University of Pittsburgh, USA
[2] Graduate School of Informatics, Middle East Technical University, Turkey
{xian.teng, yanmuheng, ertugrul, yurulin}@pitt.edu

Abstract

The increasing and flexible use of autonomous systems in many domains – from intelligent transportation systems, information systems, to business transaction management – has led to challenges in understanding the "normal" and "abnormal" behaviors of those systems. As the systems may be composed of internal states and relationships among sub-systems, it requires not only *warning* users to anomalous situations but also provides *transparency* about how the anomalies deviate from normalcy for more appropriate intervention. We propose a unified anomaly discovery framework "DeepSphere" that simultaneously meet the above two requirements – identifying the anomalous cases and further exploring the cases' anomalous structure localized in spatial and temporal context. DeepSphere leverages deep autoencoders and hypersphere learning methods, having the capability of isolating anomaly pollution and reconstructing normal behaviors. DeepSphere does not rely on human annotated samples and can generalize to unseen data. Extensive experiments on both synthetic and real datasets demonstrate the consistent and robust performance of the proposed method.

1 Introduction

An rapid growth of intelligent networked systems have been deployed in a range of environments, such as intelligent transportation systems, smart grid, cloud computing facility, and business transaction management systems. These systems are usually of high complexity since they are composed of many interdependent and time-varying components. For example, intelligent transportation systems incorporates sensors, analysis and communication technologies into vehicles and infrastructures to implement real-time traffic monitoring and information communications among sub-components.

For such networked systems, a central task is to understand the systems' normal patterns, and automatically identify anomalous behaviors, so as to conduct timely and specific interventions to guarantee system stability [Cheng *et al.*, 2016]. In particular, when system faults occur, it is required to not only deliver a *warning* signal of current system abnormality, but also provide a certain level of *transparency* regards the nested information, i.e., which components are in anomalous operation, when the anomalies occur, and how the observed measurements deviate from normal values. Significant research efforts have been devoted to anomaly detection in such dynamic networked systems (commonly described by dynamic graphs). The works in [Schölkopf *et al.*, 2000; Breunig *et al.*, 2000] are aimed at detecting anomalous graph snapshots by comparing the consecutive graphs using a distance function; however, lower-level anomalies nested in the systems are neglected (no transparency). More recent works have focused on the low-level anomaly discovery, including identifying anomalous clusters [Chen and Neill, 2014; Rozenshtein *et al.*, 2014] and individual components, i.e., nodes and edges [Teng *et al.*, 2017]. These works, however, cannot provide a preliminary judgment in terms of the entire system's performance (no warning).

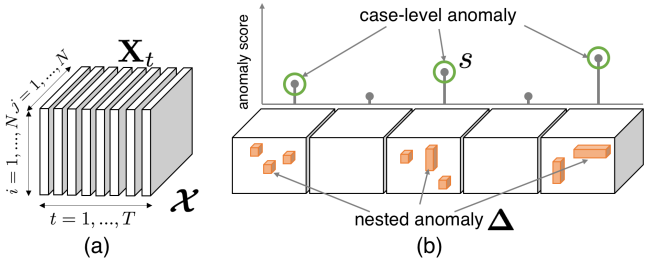

Figure 1: Illustration of the two-level anomaly discovery task.

To deal with the task mentioned above, we formulate a new "inductive and two-level anomaly discovery" problem. Firstly, the goal is two-level – we seek to learn a model that can satisfy both the *warning* and *transparency* requirements. Secondly, the model should be *inductive*, i.e., it can be generalized to unseen test data. Figure 1 illustrates the basic idea: given a set of observation samples for a dynamic graph – each represented as a tensor characterizing the internal spatio-temporal structure (Figure 1(a)), the model should inductively identify (i) anomalous sample cases and (ii) nested anomalies within the anomalous tensors (Figure 1(b)).

To resolve this problem, we propose a unified and un-

*Corresponding author.

supervised anomaly discovery framework, called "DeepSphere", that leverages deep autoencoder along with hypersphere learning technique. Deep autoencoders have demonstrated their strong capability in learning non-linear representations that capture the major patterns of input data [LeCun *et al.*, 2015]; However, the unlabelled input data are not necessarily outlier-free, i.e., it might be polluted by some unknown anomalous samples, referred as "anomaly pollution". Unfortunately, anomaly pollution might interfere learning process and therefore severely reduce the neural network's quality. We introduce hypersphere learning to overcome this problem – it allows for careful exclusion of anomaly pollution by learning a compact boundary separating normal and abnormal data points. Our unified architecture, DeepSphere, can not only perceive system fault to trigger warning signal, but also discover the specific nested anomalies within the system. The key contributions of this work include:

1. We develop a new inductive, two-level anomaly discovery problem in dynamic graphs, which is critical for appropriate management of anomalous situations in intelligent networked systems.
2. We propose a novel approach DeepSphere, which is a unified, end-to-end and unsupervised learning process that does not require either outlier-free or labeled training data. To the best of our knowledge, this is the first work that incorporates hypersphere learning into a deep learning architecture.
3. We conduct extensive experiments on both synthetic and real datasets, showing that DeepSphere outperforms the state-of-the-art baseline methods regards the two subtasks.

The paper is organized as follows. In section 2, we briefly review the relevant work. In section 3, we provide problem definition and annotations. We describe the proposed DeepSphere in section 4, and discuss our experimental results in section 5. Finally, the paper is concluded in section 6.

2 Related Work

2.1 Anomaly Detection in Dynamic Networks

Recent years have witnessed a research shift in focus to anomaly detection in dynamic graphs (rather than static graphs) [Ranshous *et al.*, 2015]. Based on application needs, detection targets can range from case-level anomaly detection to nested (lower-level) anomaly discovery. For example, GraphScope is a case-level detection method that finds the discontinuity change points in a stream of graphs [Sun *et al.*, 2007]; EventTree+, NetSpot and NPHGS are proposed to find medium-level anomalies, i.e. abnormal subgraphs or clusters [Chen and Neill, 2014; Rozenshtein *et al.*, 2014; Mongiovi *et al.*, 2013]; and Time-Series Hyperesphere Learning (TSHL) seeks to spot microscopic anomalies, i.e. nodes or edges in the graphs [Teng *et al.*, 2017]. However, none of these techniques has the capability to perform anomaly mining from both high-level and low-level perspectives. Besides, the features that have been used are also different. EventTree+ uses aggregated measurements, NPHGS relies on derived statistics, and TSHL utilizes the linear embeddings extracted from a low-dimensional space. However, they might not be suitable for capturing the complex relationship among the input features. Our DeepSphere architecture is able to learn the potential hidden nonlinear features to encode rich information from the dynamic input data, which is more efficient and effective in practical applications.

2.2 Deep Learning in Anomaly Detection

Many approaches based on deep learning techniques have been proposed for anomaly detection [Zhou and Paffenroth, 2017; Chalapathy *et al.*, 2017; Malhotra *et al.*, 2015; Malhotra *et al.*, 2016; Erfani *et al.*, 2016], we mainly review two types of neural networks used as building blocks, i.e., deep autoencoders and recurrent neural networks.

Deep Autoencoder. Deep autoencoder is an unsupervised learning architecture that has been employed in learning low-dimensional nonlinear features across many domains [LeCun *et al.*, 2015]. Recently two anomaly detection techniques have been proposed based on deep autoencoder, namely Robust Deep Autoencoder (RDA) [Zhou and Paffenroth, 2017] and Robust Convolutional Autoencoder (RCAE) [Chalapathy *et al.*, 2017]. Both methods inherit the nonlinear representation capability of autoencoder, along with the anomaly discrimination ability of robust principal component analysis (RPCA). In specific, they split the input matrix into a low-rank matrix and an outlier matrix, assuming that the outlier matrix should be sparse; besides, the input split results in a loss function that is unfortunately not computationally tractable. Whereas DeepSphere does not have any requirements towards anomaly sparsity; more importantly, it is a unified end-to-end learning model which does not require any optimization algorithm development.

Recurrent Neural Networks. Recurrent neural network (RNNs) are artificial neural network designed for processing time-series and sequence data [Liu *et al.*, 2016]. Therefore, it becomes a natural choice to be used for developing new methods to detect anomalies in time-series data. For example, Malhotra et al. develop two methods, i.e. LSTM-AD [Malhotra *et al.*, 2015] and EncDec-AD [Malhotra *et al.*, 2016], based on Long Short-Term Memory (LSTM) for anomaly detection in time series data. However, both models are supervised methods which require clean normal inputs in training process. Whereas we attempt to train an unsupervised model to cope with the lack-of-label and potentially polluted data. DeepSphere achieves this goal adding a hypersphere learning component to preclude outliers and accordingly maintain high quality reconstruction of normal behaviors.

3 Problem Definition

In this section, we introduce definitions, notations and problem formulation.

Definition 3.1. **Dynamic Graph.** A dynamic graph is defined as $G(t) = \{\mathrm{V}, \mathrm{E}, x(t)\}$, where V denotes vertex set (with number N), E denotes edge set, and $x(t)$ is a mapping function that associates each edge e_{ij} with a time series $\{x_{ij}(t),\ t = 1, ..., T\}$. As shown in Figure 1(a), one observation case of $\mathrm{G}(t)$ can be described by a three-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N \times N \times T}$, and the lateral slices along time dimension

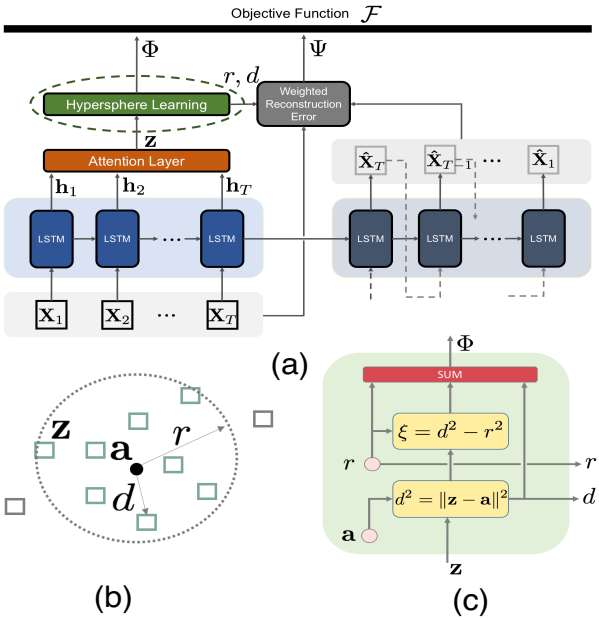

Figure 2: Illustration of DeepSphere architecture.

are the graph's adjacency matrices at different time steps, denoted as $\{\mathbf{X}_t, t = 1, ..., T\}$. A collection of cases can be denoted as $\{\boldsymbol{\mathcal{X}}_k, k = 1, 2, ...\}$.

Definition 3.2. **Inductive & Two-level Anomaly Discovery.** For a dynamic graph $\mathrm{G}(t)$, assuming that there is a set of historical observations (i.e., the training data) $\{\boldsymbol{\mathcal{X}}_k, k = 1, ..., m\}$, we determine to train a model based on the training data, and then inductively apply the trained model to unseen data (i.e., test data) $\{\boldsymbol{\mathcal{X}}_k, k > m\}$. In particular, the task is of two levels (Figure 1(b)): (i) *Case-level anomaly detection*: identifying which observation cases (i.e., tensors) in the test data, say $\{\boldsymbol{\mathcal{X}}_u, u > k\} \subset \{\boldsymbol{\mathcal{X}}_k, k > m\}$), are anomalous; (ii) *Nested anomaly discovery*: discovering the anomalous cells nested within the anomalous tensors in test data, and calculate how they deviate from the expected normal behaviors.

Actually, the first task is aimed at dealing with the *warning* requirement: an anomaly score $s(\boldsymbol{\mathcal{X}}_k)$ is computed for each case $\boldsymbol{\mathcal{X}}_k$ conveying a signal whether the system is likely to be normal or not; the second task is aimed at providing some *transparency*: a "difference" tensor $\boldsymbol{\Delta}(\boldsymbol{\mathcal{X}}_k)$ would be computed, describing how the anomalous tensor deviate from the expected normal tensor.

4 Method

In this section, we first introduce some preliminary knowledge, and then describe our approach DeepSphere.

4.1 Preliminary

Hypersphere Learning. Hypersphere learning is originally proposed in the problem of data description (i.e. the characterization of a dataset) [Tax and Duin, 2004]. A good description is defined as a compact boundary (also called *hypersphere*) that covers all normal data points but includes no superfluous space. By learning such a compact hypersphere, outliers can be detected and precluded. As shown in Figure 2(b), the hypersphere can be characterized by a centroid $\mathbf{a}$ and a radius r, and the set of data points is denoted as $\{\mathbf{z}_k, k = 1, ..., m\}$. The error function that needs to be minimized is:

$$\Phi(\mathbf{a}, r) = r^2 + \gamma \sum_k \xi_k, \tag{1}$$

with the constraints:

$$\|\mathbf{z}_k - \mathbf{a}\|^2 \leq r^2 + \xi_k, \xi_k \geq 0, \quad \forall i, \tag{2}$$

where ξ_k are slack variables to allow the possibility of anomalies in the dataset. The distance from $\mathbf{z}_k$ to $\mathbf{a}$ is not strictly smaller than r^2 but larger distance should be penalized (the data points outside the boundary are anomaly pollution). Besides, the parameter γ controls the trade-off between sphere volume and penalization. By minimizing Eq.1, the parameters, i.e. centrold $\mathbf{a}$ and radius r, can be obtained (note that ξ_k is not part of the parameter set because it is an auxiliary variable rather than the learning goal). Our proposed DeepSphere model is equipped with a hypersphere learning component which plays a critical role in isolating and penalizing anomaly pollution contained in the input data.

LSTM Autoencoder. Deep autoencoders usually contain two parts: the encoder $\alpha_{\boldsymbol{\theta}}$ and the decoder $\beta_{\boldsymbol{\phi}}$, which are nonlinear mapping functions implemented via neural networks with parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, respectively. The encoder maps input data into the low-dimensional hidden layer $\mathbf{z} = \alpha_{\boldsymbol{\theta}}(\mathbf{X})$, while the decoder maps from the hidden layer into the output layer to reconstruct input $\hat{\mathbf{X}} = \beta_{\boldsymbol{\phi}}(\mathbf{z})$. The encoder $\alpha_{\boldsymbol{\theta}}$ and decoder $\beta_{\boldsymbol{\phi}}$ can be implemented by different types of neural networks, such as feedforward non-recurrent neural networks or recurrent neural networks. Our DeepSphere is built on LSTM recurrent neural networks (Figure 2(a)), in order to better capture the potential temporal dependency and the structural relationship within dynamic graphs.

4.2 The Proposed DeepSphere

Motivation. To resolve our two-level research problem, we derive our method DeepSphere by integrating autoencoders with hypersphere learning in a mutual supportive manner. On one hand, DeepSphere inherits the anomaly separation capability of hypersphere learning, helping to improve the quality of autoencoders (more dedicated to the normal patterns); on the other hand, DeepSphere bears the advantages of autoencoders – being able to capture the spatio-temporal dependencies among components and across timesteps, to flexibly learn a nonlinear feature representation, and to reconstruct the normal behaviors from potentially anomalous input data. The high-quality nonlinear representations learned by autoencoder, in turn, helps hypersphere learning to better distinguish anomalous cases. Next we will show that DeepSphere combines the two parts by designing hypersphere learning as a special layer (parameterized by $\mathbf{a}$ and r) in a unified deep neural network architecture, without necessity of developing any ad-hoc learning mechanism.

DeepSphere Architecture. Figure 2(a) shows the overall DeepSphere architecture. A sample case $\boldsymbol{\mathcal{X}}$ is sliced into a

series of matrices $\{\mathbf{X}_t, t = 1, ..., T\}$ corresponding to a sequence of graphs. They are fed into a LSTM encoder and a series of internal states $\{\mathbf{h}_t, t = 1, ..., T\}$ can be generated. The $\{\mathbf{h}_t\}$ captures the information about the source sequence $\{\mathbf{X}_t\}$, including short- and long-range dependencies. To learn a more flexible dependency structure, we employ the attention mechanism to allocate various attention to different $\{\mathbf{h}_t\}$ [Denil *et al.*, 2012], i.e., $\mathbf{z} = \sum_t \omega_t \mathbf{h}_t$, where $\mathbf{z}$ is the embedded representation, and ω_t represents attention weight at time step t.

In the hidden space, we design a new layer called "hypersphere learning" which learns a spherically shaped boundary around the encoded representations $\{\mathbf{z}_k\}$ to tell apart of anomaly pollution (Figure 2(b)). Specifically, the inner structure of hypersphere learning layer (along with its inputs and outputs) is displayed in Figure 2(c). The input is $\{\mathbf{z}_k\}$, the two parameters r and $\mathbf{a}$ are denoted as nodes, the functions used to calculate the distance d and outlier penalty ξ are taken as two nonlinear neurons. To minimize the chance of accepting anomalous cases, the objective function is defined as:

$$\Phi = r^2 + \gamma \sum_{k=1}^{m} \xi_k + \frac{1}{m} \sum_{k=1}^{m} \|\mathbf{z}_k - \mathbf{a}\|^2. \tag{3}$$

Here we strictly assume that all normal tensors should be mapped onto the centroid $\mathbf{a}$ of the hypersphere, therefore we add a new item (the 3rd item) minimizing the average distance between $\mathbf{z}_k$ and $\mathbf{a}$. Finally, the hypersphere learning layer outputs Φ, d and r.

In the above learning process, the latent representations $\mathbf{z}$ lying outside of the hypersphere with large distances tend to be anomalous, while the ones lying inside of the hypersphere of small distances are prone to be normal ones. We utilize this knowledge to modify the reconstruction error for LSTM autoencoder as follows (corresponding to "weighted reconstruction error" in Figure 2(a)):

$$\Psi = \sum_{k=1}^{m} \eta_k \|\boldsymbol{\mathcal{X}}_k - \hat{\boldsymbol{\mathcal{X}}}_k\|^2, \tag{4}$$

where $\boldsymbol{\mathcal{X}}_k$ is reconstructed through LSTM decoder, and the η_k indicates case-wise weights computed through a heuristic function $\eta(d_k, r)$. It should reward the latent representations $\mathbf{z}$ lying near to $\mathbf{a}$, while penalize (or even remove) anomalous cases outside of the hypersphere. Please note that different types of functions can be chosen for $\eta(d_k, r)$ as long as it distinguishes $\mathbf{z}$ based on the boundary. For example, the following function can be used:

$$\eta(d_k, r) = \frac{1}{\tau}[\max(\exp(r^2 - d_k^2) - 1, 0) + \epsilon], \tag{5}$$

where τ is the normalization parameter $\tau = \sum_j \eta(d_j, r)$, and ϵ is a considerably small value. It implies that the cases lying outside of hypersphere are removed, while those inside hypersphere hold exponential weights in reconstruction stage.

In summary, the overall objective function is the combination of the hypersphere component Φ and the panalized reconstruction difference Ψ:

$$\min_{\boldsymbol{\Theta}} \mathcal{F} = \min_{\boldsymbol{\Theta}} \{\Phi + \lambda \Psi\} \tag{6}$$

where λ is the trade-off parameter between these two items, and $\boldsymbol{\Theta} = \{\mathbf{a}, r, \boldsymbol{\omega}, \boldsymbol{\theta}, \boldsymbol{\phi}\}$ is the parameter set containing the hypersphere centroid $\mathbf{a}$, the radius r, the attention parameter $\boldsymbol{\omega}$, as well as the neural network parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$ for LSTM encoder and decoder. According to Eq.6, if λ is extremely large, making hypersphere learning vanished, the model would degenerate into an autoencoder (lose the ability to filter out outliers); if λ is extremely small, making reconstruction process vanished, it would become a method particularly for case-level detection (lose the ability to reconstruct normal pattern).

Training and Testing. Since every part of the DeepSphere architecture is differentiable, the model can be trained in an end-to-end manner by using the back propagation. In our implementation, we select the Adam Optimizer to train our model [Kingma and Ba, 2014]. As DeepSphere has been trained, given a new unseen sample $\boldsymbol{\mathcal{X}}_k (k > m)$, we can perform case-level anomaly detection by examining its distance towards $\mathbf{a}$ (taken as anomaly score $s(\mathbf{X}_k)$) and make decision accordingly. Besides, DeepSphere is able to reconstruct its normal behavior $\hat{\boldsymbol{\mathcal{X}}}_k$ even though $\boldsymbol{\mathcal{X}}_k$ is an anomalous input. By computing the reconstruction difference $\boldsymbol{\Delta}(\boldsymbol{\mathcal{X}}_k) = \boldsymbol{\mathcal{X}}_k - \hat{\boldsymbol{\mathcal{X}}}_k$, we can discover the nested anomalies localized in temporal and spatial dimensions.

5 Experiments

5.1 Datasets

Synthetic Data. We generate synthetic data (ground truth can be known) in order to conduct comprehensive performance evaluation. Particularly, we will implement it from the following four perspectives: (1) *anomaly pollution*: since there is no guarantee for clean training data (i.e., it may be polluted by anomalies), the level of anomaly pollution can affect DeepSphere's performance; (2) *spatio-temporal locality*: anomalies usually appear clustered spatially or temporally. The variation between spatial locality or temporal locality could affect the detection; (3) *nested anomaly extent*: the extent to which a data sample contains nested anomalies could affect the detection; (4) *data imbalance*: the extent to which the testing data contain anomaly cases can affect the detection. We thus design the following process to simulate these different conditions.

The data is generated by three steps: (i) randomly create a graph structure, (ii) assign normal behaviors, and (iii) insert anomalies. In step (iii), we first determine the fraction of case-level anomalous tensors within the data (corresponding to *anomaly pollution* in training data, and *data imbalance* in test data), and then plant the nested anomalous cells localized inside the tensor $\boldsymbol{\mathcal{X}}$. To guarantee spatial or temporal locality, a random walk (RW) is employed to select the target cells in the tensor. The walker starts from an arbitrary cell (i, j, t), with a probability p it would stay at the current time step t (walking over graph), or with a chance of $1-p$ it would jump to the previous or the next time steps. The parameter p controls the *spatio-temporal locality* – traversing more over

graph or more across time. The walker would stop when satisfying a predefined stop condition, which is characterized by the fraction of visited cells in the tensor. RW stop condition controls the *nested anomalies extent*.

Real-word Data. To demonstrate DeepSphere's application in diverse domains, we apply it on three realistic datasets.

A. New York City (NYC) Taxi Trips. The NYC taxi trip data[1] is a public transportation dataset. Each piece of record includes a trip's detailed information, such as pick up time, pick up location, drop off time and drop off location etc. We extract a subset of trips (more than 41 million) from July 2016 to December 2016 and construct a transportation graph. In this graph, vertices represent zones (e.g. Midtown Center and Upper East Side) and edges represent taxi transportation. We remove the insignificant zones with few number of trips, and finally obtain 48 zones left. We can detect anomalous transportation phenomena by mining this dataset.

B. Tweets & News Collection. We use a semantic dataset called "HERMEVENT" which was released in this paper [Crescenzo *et al.*, 2017]. HERMEVENT is created from a large corpus containing tweets and news articles collected from both Twitter and major Italian news papers, spanning from December 2016 to March 2017. Based on the corpus, the authors build a set of temporal graphs where the vertices represent entities (i.e. phrases) and the edges encode the co-occurrence frequency of two phrases. In this paper, the Italian phrases are translated into English for the convenience of understanding; and we extract a subgraph (containing 160 entities) as our analysis target. We rely on this data to identify reported news that correspond to events happened in the real world.

C. Restaurant Video. The restaurant video data is a set of frames (images) captured in a restaurant, which has been used for background modeling and foreground activity detection [Chalapathy *et al.*, 2017]. Background means the relatively static scenes, while foreground activity might be customers coming, talking at reception and leaving. Unlike the general foreground-background separation task in computer vision research [Braham and Droogenbroeck, 2016], here we consider the problem of detecting foreground activities without using additional image processing or background learning.

5.2 Baseline Methods and Evaluation Metrics

We compare our DeepSphere model with the following approaches: Robust Autoencoder (RAE) [Zhou and Paffenroth, 2017], LSTM Autoencoder (LSTM-AE) [Baytas *et al.*, 2017], Time-Series Hyperesphere Learning (TSHL) [Teng *et al.*, 2017], Local Outlier Factor (LOF) [Breunig *et al.*, 2000], and One-Class SVM [Schölkopf *et al.*, 2000]. DeepSphere, RAE and LSTM-AE are deep learning based methods that perform nonlinear representation learning and can do nested anomaly discovery; TSHL is a linear technique also equipped with hypersphere learning component; LOF and One-Class SVM are two domain-independent approaches. For RAE, we have modified the base version to be a LSTM autoencoder in order to make it compatible with the time series data.

[1]http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

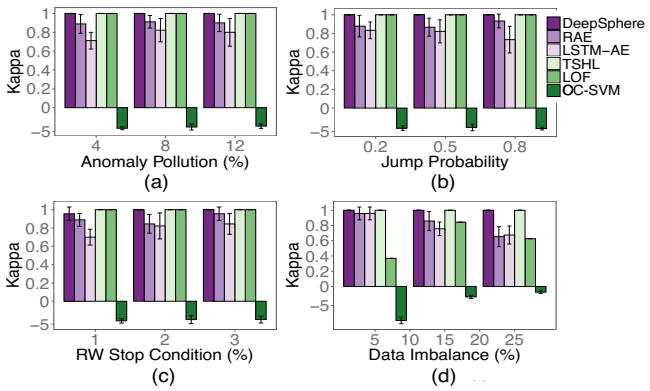

Figure 3: Results of case-level anomaly detection described by Kappa statistics.

We evaluate the methods' performance in terms of two tasks: (i) case-level anomaly detection, and (ii) nested anomaly discovery. For the first task, we employ the Kappa statistics as the evaluation metric: $\kappa = (acc - acc_{exp})/(1 - acc_{exp})$ to account for data imbalance effect. Here acc is the calculated accuracy, while acc_{exp} is the expected accuracy. For the second task, we use root mean square error: $\text{RMSE} = \sqrt{\text{E}(\hat{\mathbf{\Delta}} - \mathbf{\Delta})^2}$. Here $\hat{\mathbf{\Delta}}$ is the approximated nested anomalies given by DeepSphere, and $\mathbf{\Delta}$ is the ground truth difference injected in simulation process[2].

5.3 Experimental Results.

Performance Comparison on Synthetic Data.

Figure 3 shows the performance comparison of different methods in detecting case-level anomalies, in terms of the four aspects mentioned above. The two hypersphere learning based approaches, DeepSphere and TSHL, perform the best – both can detect almost all case-level anomalies under all conditions. This is expected as the hypersphere learning component has better capability in dealing with anomaly pollution in training data. LOF exhibits similar good performance, but its performance declines drastically when changing the level of data imbalance (Figure 3(d)). Besides, One-Class SVM displays negative Kappa values, implying that its performance in the experiments is worse than random guessing. A possible explanation for its poor performance is the curse of dimensionality: One-Class SVM is demonstrated to be very inefficient in producing decision surfaces in high dimensional feature spaces [Khan and Madden, 2009; Erfani *et al.*, 2016]. Comparing DeepSphere with the other two deep learning models, we can observe that our proposed DeepSphere shows consistent better performance across different scenarios.

Figure 4 provides the performance of three deep learning models (i.e. DeepSphere, RAE, LSTM-AE) in discovering nested anomalies, as TSHL, LOF, and One-Class SVM cannot perform this task. We observe that DeepSphere gives the lowest RMSE values in all scenarios. It maintains stable performance when varying anomaly pollution

[2]Code is available at https://github.com/picsolab/DeepSphere

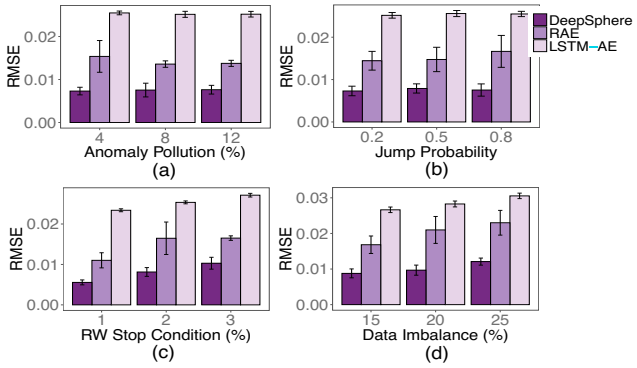

Figure 4: Results of nested anomaly discovery described by RMSE.

and jump probability (Figure 4(a)(b)). DeepSphere's performance slightly declines when the size of nested anomalies increases (Figure 4(c)), and when the data imbalance increases (Figure 4(d)) – in both scenarios, fully capturing the nested anomalies would be more challenging. In summary, in our experiments DeepSphere shows a superior performance in the nested anomaly discovery task, and holds comparably the best performance in the case-level detection task. The results also indicate that DeepSphere is more robust against the changing nature of anomalies in the training data (e.g., anomaly pollution, spatio-temporal locality, nested anomaly extent) or in the unseen data (data imbalance).

Case Studies on Real-World Data.

A. NYC Taxi Trips. Figure 5 shows the taxi transportation snapshots for four consecutive time periods during Thanksgiving day on November 24th, 2016 in Manhattan. Specifically, Figure 5(a)(b) shows the taxi traffic flow under normal situation and during Thanksgiving respectively. Hue indicates the level of incoming flow for the corresponding regions. We can easily observe that there is a decline in taxi transportation on Thanksgiving, especially for the busiest regions (e.g. "Upper East Side" and "Midtown Manhattan"). The ground truth change is reported in Figure 5(c), and our detection results are provided in Figure 5(d). By comparing (c) and (d), we can see that our proposed DeepSphere model is able to uncover the nested decline in taxi flows localized at various time steps and in different sub-regions.

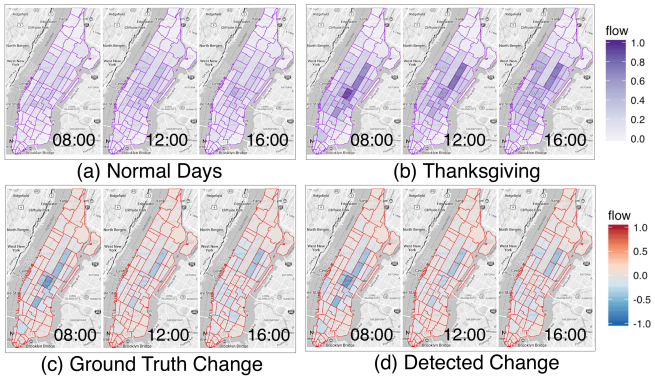

Figure 5: Decreased taxi trips on Thanksgiving day in Manhattan.

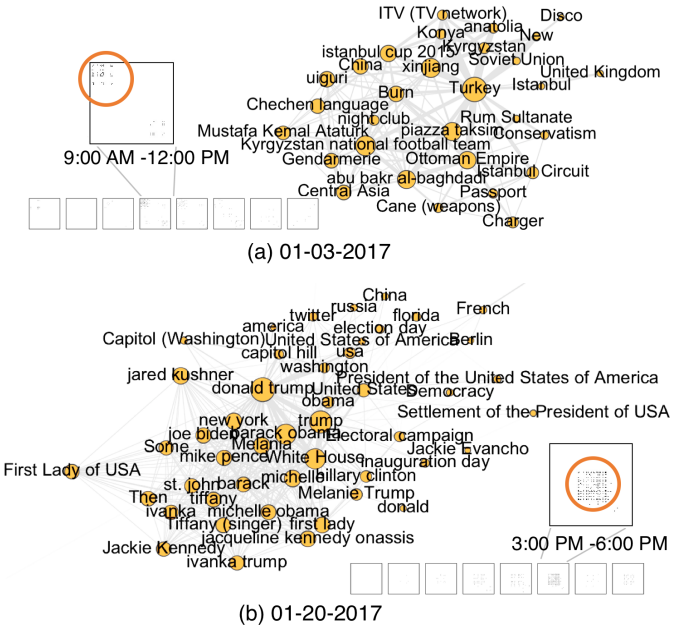

Figure 6: Two events detected in HERMEVENT data. Italian has been translated into English.

B. Tweets & News Collection. Figure 6 provides two events detected by DeepSphere in HERMEVENT temporal graphs: (a) the Kyrgyz man identified as the nightclub shooter in Istanbul, on January 3rd, 2017; and (b) the inauguration of Donald Trump as the 45th President of USA in Washington, D.C. on January 20th, 2017. Each day contains eight snapshots showing the nested anomalies captured by DeepSphere. By zooming the most anomalous one (9:00 AM - 12:00 PM), we can find a dotted cluster on the upper left corner. This cluster suggests that a group of phrases has co-appeared frequently on that day. To examine what the phrases are, we also plot the node-edge diagram associated with the cluster. Referring to this news article[3], we can find that the phrases – *Istanbul*, *Turkey*, *Kyrgyz*, *Xinjiang* and *Uighurs* – are closely correlated with this event. In a similar way, we can detect a compact cluster as the evidence of the second event. The associated node-edge diagram is also formed by many informative phrases, such as *Settlement of the President of USA*, *inauguration day*, as well as relevant names – *Donald Trump*, *Hillary Clinton*, *Barack Obama*, *Michelle Obama*, *Jared Kushner*, *Mike Pence*, etc.

C. Restaurant Video. Figure 7 reports the activity detection results in the restaurant video data. Two activity instances – people coming and leaving – are provided. The top, middle and bottom rows represent normal situation, anomalous situation and detected results separately. Without additional image processing steps, we can see that the foreground activities are well captured by DeepSphere. The result suggests that the DeepSphere has a broad application with tasks similar to the anomaly discovery in dynamic graphs.

[3] http://palermo.gds.it/2018/01/31/spari-su-un-autobus-a-palermo-proiettile-sul-vetro-nessun-ferito_795630/

(a) (b)

Figure 7: Two examples of foreground activity detection in restaurant video.

6 Conclusions

We propose a two-level inductive anomaly discovery task in dynamic graphs, and develop a novel DeepSphere framework based on deep autoencoders and hypersphere learning. DeepSphere demonstrated better and robust performance than baseline methods, and also exhibits a broad application in a variety of domains (partially demonstrated by the real-world experiments). As part of future work, we plan to (i) extend our framework to exploit multiple data sources; and (ii) explore the problem of anomaly source detection in networks.

Acknowledgements

This work is part of the research supported from NSF #1634944, #1637067, and #1739413.

References

[Baytas *et al.*, 2017] Inci M. Baytas, Cao Xiao, et al. Patient subtyping via time-aware lstm networks. In *SIGKDD*, pages 65–74. ACM, 2017.

[Braham and Droogenbroeck, 2016] Marc Braham and Marc Van Droogenbroeck. Deep background subtraction with scene-specific convolutional neural networks. In *IWSSIP*, pages 1–4. IEEE, 2016.

[Breunig *et al.*, 2000] Markus M. Breunig, Hans-Peter Kriegel, et al. Lof: identifying density-based local outliers. In *SIGMOD*, volume 29, pages 93–104. ACM, 2000.

[Chalapathy *et al.*, 2017] Raghavendra Chalapathy, Aditya K. Menon, and Sanjay Chawla. Robust, deep and inductive anomaly detection. *arXiv preprint arXiv:1704.06743*, 2017.

[Chen and Neill, 2014] Feng Chen and Daniel B. Neill. Non-parametric scan statistics for event detection and forecasting in heterogeneous social media graphs. In *SIGKDD*, pages 1166–1175. ACM, 2014.

[Cheng *et al.*, 2016] Wei Cheng, Kai Zhang, et al. Ranking causal anomalies via temporal and dynamical analysis on vanishing correlations. In *SIGKDD*, pages 805–814. ACM, 2016.

[Crescenzo *et al.*, 2017] Cristiano Di Crescenzo, Giulia Gavazzi, et al. Hermevent: a news collection for emerging-event detection. In *WIMS*, page 11. ACM, 2017.

[Denil *et al.*, 2012] Misha Denil, Loris Bazzani, et al. Learning where to attend with deep architectures for image tracking. *Neural Computation*, 24(8):2151–2184, 2012.

[Erfani *et al.*, 2016] Sarah M. Erfani, Sutharshan Rajasegarar, et al. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, 58:121–134, 2016.

[Khan and Madden, 2009] Shehroz S. Khan and Michael G. Madden. A survey of recent trends in one class classification. In *AICS*, pages 188–197. Springer, 2009.

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

[LeCun *et al.*, 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[Liu *et al.*, 2016] Jun Liu, Amir Shahroudy, et al. Spatio-temporal lstm with trust gates for 3d human action recognition. In *ECCV*, pages 816–833. Springer, 2016.

[Malhotra *et al.*, 2015] Pankaj Malhotra, Lovekesh Vig, et al. Long short term memory networks for anomaly detection in time series. In *Proceedings*, page 89. Presses universitaires de Louvain, 2015.

[Malhotra *et al.*, 2016] Pankaj Malhotra, Anusha Ramakrishnan, et al. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.

[Mongiovi *et al.*, 2013] Misael Mongiovi, Petko Bogdanov, et al. Netspot: Spotting significant anomalous regions on dynamic networks. In *ICDM*, pages 28–36. SIAM, 2013.

[Ranshous *et al.*, 2015] Stephen Ranshous, Shitian Shen, et al. Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(3):223–247, 2015.

[Rozenshtein *et al.*, 2014] Polina Rozenshtein, Aris Anagnostopoulos, et al. Event detection in activity networks. In *SIGKDD*, pages 1176–1185. ACM, 2014.

[Schölkopf *et al.*, 2000] Bernhard Schölkopf, Robert Williamson, et al. Support vector method for novelty detection. In *NIPS*, pages 582–588, 2000.

[Sun *et al.*, 2007] Jimeng Sun, Spiros Papadimitriou, et al. Graphscope: parameter-free mining of large time-evolving graphs. In *SIGKDD*, pages 687–696. ACM, 2007.

[Tax and Duin, 2004] David M.J. Tax and Robert P.W. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

[Teng *et al.*, 2017] Xian Teng, Yu-Ru Lin, and Xidao Wen. Anomaly detection in dynamic networks using multi-view time-series hypersphere learning. In *CIKM*, pages 827–836. ACM, 2017.

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