Event2Graph: Event-driven Bipartite Graph for Multivariate Time Series Forecasting and Anomaly Detection

Yuhang Wu[1,*], Mengting Gu[1], Lan Wang[1], Yu-san Lin[1], Fei Wang[1] and Hao Yang[1]

[1]Visa Research, 385 Sherman Ave., Palo Alto, CA, U.S.A

Abstract

Modeling inter-dependencies between time series is the key to achieve high performance in anomaly detection for multivariate time series data. The de-facto solution to model the dependencies is to feed the data into a recurrent neural network (RNN). However, the fully connected network structure underneath the RNN (either GRU or LSTM) assumes a static and complete dependency graph between time series, which may not hold in many real-world applications. To alleviate this assumption, we propose a dynamic bipartite graph structure to encode the inter-dependencies between time series. More concretely, we model time series as one type of nodes, and the time series segments (regarded as event) as another type of nodes, where the edge between two types of nodes describe a temporal pattern occurred on a specific time series at a certain time. Based on this design, relations between time series can be explicitly modeled via dynamic connections to event nodes, and the multivariate time series anomaly detection problem can be formulated as a self-supervised, edge stream prediction problem in dynamic graphs. We conducted extensive experiments to demonstrate the effectiveness of the design.

Keywords

Time series, Anomaly detection, Dynamic graph,

1. Introduction

Detecting anomalies in time series data has been an important problem in the research community of data mining as well as the finance industry. In many circumstances, anomaly patterns in multiple time series need to be taken into account together to disclose the full picture of the system. Previous works in multivariate anomaly detection mainly rely on recurrent neural networks (RNNs). Malhotra *et al.*[1] and Hundman *et al.*[2] employed LSTM models [3] to capture the temporal dependencies of multivariate signals and adopted prediction and reconstruction errors, respectively, to identify the anomaly patterns. Su *et al.*[4] improved the classical RNN model by modeling the probability distribution of time series via incorporating stochastic variables. To further model the correlations of time series explicitly, Zhao *et al.*[5] proposed a graph attention network to propagate information from different time series and aggregate the information together before feeding into a GRU [6]. However, because of the underlying RNN structure, previous works assume a static, complete dependency graph among time series. These approaches may not perform well under regime change of time series where the underlying inter-dependencies are different. Our work is built upon the recent wisdom of dynamic graph neural network. We allow the connectivity of the graph changes dynamically in different time stamps based on the patterns on the time series.

In order to construct a dynamic graph on-the-fly, one important question is how to determine the connectivity of the graph in each timestamp. Our work is inspired by the recent progress of evolutionary event graph [7, 8] where the nodes in the graph represent the time-sequences segments (events) and directed links represent the transition of the segments (events). Compare to previous works, this line of research naturally models the time-varying relations among time series states via dynamic connections, and each state carries a physical meaning that is understandable by human. However, one major limitation of [8] is that the event nodes employed in this work capture the information across all the time series. Assume there are K segment patterns in each time series, and the number of time series is D. The model would need K^D number of event nodes to represent the multivariate signals. This exponential number of event nodes strongly limits the information processing capability of the evolutionary state graph and therefore allows only a very small number of time series (D) to be analyzed in practice (four in the dataset used in the paper [8]).

To address the problem of exponential number of combinations, we disentangle the time series nodes and the event nodes in our design, and model them as two types of nodes in a dynamic bipartite graph (as depicted in Fig. 1). Each event node only represents a time segment on one individual time series, instead of integrating patterns across all time series. The undirectional con-

AMLTS'22: Workshop on Applied Machine Learning Methods for Time Series Forecasting, co-located with the 31st ACM International Conference on Information and Knowledge Management (CIKM), October 17-21, 2022, Atlanta, USA

[*]Corresponding author.

yuhawu@visa.com (Y. Wu); mengu@visa.com (M. Gu); lwang4@visa.com (L. Wang); yusalin@visa.com (Y. Lin); feiwang@visa.com (F. Wang); haoyang@visa.com (H. Yang)

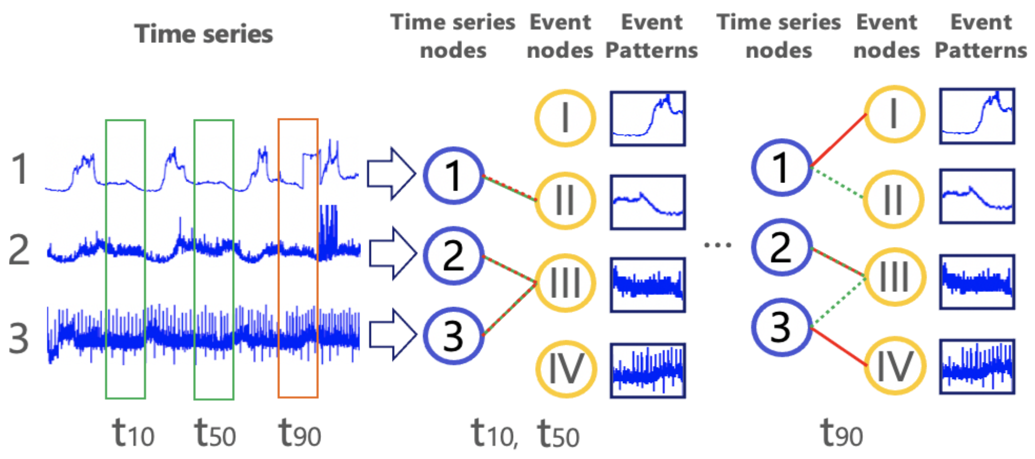

Figure 1: This toy example has three time series nodes and four event nodes. Time-step $\mathbf{t}_{90}$ correspond to an anomaly pattern while other time stamps are normal. The relation between time series nodes and event nodes are highlighted, green dashed line means the predicted relation while the red line means the actual relation.

nection between two types of nodes indicates event e happens on the d^{th} time series at time t. So the maximum number of edges in the graph is $O(KD)$, which is much smaller than $O(K^{2D})$. To further improve the efficiency and generalizability of the algorithm, we built upon the framework based on the recent advances in edge streams [9, 10, 11], where connections between nodes are modeled as incoming attributed edges instead of constructing adjacency matrices. The complete system, with the name Event2Graph (Event-driven bipartite Graph), outperforms previous strong baselines on two public data-sets, and we summarize our main contributions as follows:
(i) We propose a bipartite event-graph-based system to analyze the multivariate time series and model the interactions between time series and event segments via edge streams. (ii) Events employed in the system is highly interpretable and easy for human to interact with. (iii) The system achieves competitive performance on challenging anomaly detection datasets through graph edge forecasting.

2. Related work

Static inter-dependency relation: In this category of approach, relations (*e.g.*, correlation) between multivariate time series are fixed once learned, and the model assumes all the time series can influence each other (complete inter-dependency). LSTM-based framework has been widely employed in this category of work. For instance, Malhotra *et al.*[1] proposed a LSTM-based auto-encoder network to detect anomalies from multiple sensors, Su *et al.*[4] proposed a stochastic LSTM framework to model the data distribution of time series in a global perspective. As RNN was originally proposed to model the temporal dependencies between different timestamps, and all the dimensions of the input of RNN are used to describe a single concept (*e.g.*word) all together, it was not tailored to model the inter-dependencies among variables by design. Recently, Zhao *et al.*[5] proposed a graph attention network-based (GAT) approach [12] where each time series is regarded as an individual node, and information are aggregated based on the underlying similarity of the signals. While this solution partially mitigates the problem by taking into account of dynamic pairwise similarities between time series, the design assumes a complete and static inter-dependency graph. Also, the processed information after GAT is simply aggregated all together and fed into a single GRU. Hence the module still suffers from similar problems as previous designs.
Dynamic inter-dependency relation: A few recent works started to explore the partial inter-dependency relations among time series. Deng *et al.*[13] constructed a neighborhood graph on-the-fly based on sensor embeddings to describe the dependencies between different sensors. The sparsity level of this incomplete graph can be customized by users, but the connectivity of the graph is fixed once constructed. Hu *et al.*[8] proposed the first dynamic dependency design in multivariate anomaly detection. In this design, nodes are interpretable time series states, and the transition of these states are explicitly modeled by dynamic graph neural network. This design allows the system to represent the time-varying relations among time series. However, one major limitation of

the method is it uses a single event node to represent segments across all time series. This design brings the combination explosion problem in pattern representation (mentioned in Section I) and makes it hard to tackle a moderate scale problem. Our model is built upon the advances in dynamic graph neural networks. Most of the literatures in dynamic graph networks assume discrete-time graph dynamics where the graphs are represented as a sequence of snapshots [14, 15, 16]. Recent works start exploring more flexible design and assuming edges can appear at any time [10, 17, 9, 18, 11]. Our solution is derived from a recent solution named Temporal Graph Networks (TGN) proposed by Rossi *et al.*[11]. The model is built upon the temporal graph attention network (TGAT) [10] but extended with an unique node-wise memory which attempt to model the temporal dependency on the node level instead of the system level as [5]. It allows us to model the dynamic inter-dependency in our bi-partite event graph accurately, and provide us the flexibility to incorporate new nodes that have not seen in training.

3. Problem definition

A multivariate time series is defined as $\mathbf{X} \in \mathbb{R}^{T \times d}$ where T is the maximum number of time stamps in the time series and d is the dimension. The anomaly detection problem is to detect specific time stamps $\dot{t}^* \in \mathbb{O}$ in a multivariate time series where the time series behavior deviating from the normal patterns of the time series. Note that $\mathbb{O}$ contains timestamps that marked as anomaly by domain expert. In case T is large, a common way to handle long time series is to use a sliding window approach. Let τ denote the length of the sliding window. We hereby formulate the problem as a binary classification problem with the objective to identify time windows with $\mathbf{X}^{[\dot{t}-\tau:\dot{t}]\times d}$ that contain anomaly time-stamps. One solution of solving the multivariate time series anomaly detection problem is to convert time series into a homogeneous graph [7, 8], where $\mathbb{G}_e$ is defined on the representative patterns of a multivariate time series. The node of the graph $\mathbf{v}_e$ corresponds to a human interpretable time series patterns $\mathbf{p} \in \mathbb{R}^{\tau \times d}$. The pattern should be representative enough so that the whole time series can be approximated by transitions of symbolic events (states) in the graph. The transitional relation between two sequential nodes (end with stamp t and $t+1$) is defined as an edge in the graph. Many existing works (*e.g.*, Bag of Patterns, Shaples, sequence clustering) can be used to distill representative patterns (event) from time series, but most of them only effective on univariate time series where $d = 1$.

4. Bipartite Event Stream

The core intuition behind the dynamic bipartite design is to decouple three key concepts in the traditional event graph, namely: where (which time series), when (at what time), and which (the event category). This practice helps us avoid the problem of exponential pattern combinations.

Specifically, the proposed algorithm is built upon a dynamic bipartite graph. We define it as a sequence of undirected bipartite event graph $G_B^t = \{(\mathbf{v}_m^t, \mathbf{v}_e^t, A(\mathbf{v}_m^t, \mathbf{v}_e^t))\}$. Here t represents when the time window that the event graph is formulated. If the window size is set to be 1, then the time window t is essentially the same as the time step $\dot{t}$. A time sequence index node $\mathbf{v}_m^t$ indicates "where" an event $\mathbf{v}_e^t$ is happening. An attributed edge $A(\mathbf{v}_m^t, \mathbf{v}_e^t)$ that connect event node $\mathbf{v}_e^t$ and sequence index node $\mathbf{v}_m^t$ indicates an event $\mathbf{v}_e^t$ happened on time series m at time window t. For simplicity, we also denote the edge as $A_{m,e}$ in the following sections. We employ an edge stream representation so an edge $A(\mathbf{v}_m^t, \mathbf{v}_e^t)$ is only constructed to represent the relation that actually existed. A major benefit of using edge stream representation over adjacency matrix is that it allows the graph structure to be more scalable and flexible, which provides us the generality of incorporating new events that have not appeared in training. Based on this bipartite graph structure, we convert the multivariate anomaly detection problem into a self-supervised edge prediction problem in dynamic graph. Given the historical sequence of $\mathbb{G}_B^{(1:t)}$ and event nodes $\mathbf{v}_e^{t+1}$ in time window $t+1$, we are predicting edges $\hat{A}(\mathbf{v}_m^{t+1}, \mathbf{v}_e^{t+1}))$ in $\mathbb{G}_B^{t+1}$. The anomaly is derived as a mismatching score from the predicted edge set $\hat{A}(\mathbf{v}_m^{t+1}, \mathbf{v}_e^{t+1}))$ and the observed edge set $A(\mathbf{v}_m^{t+1}, \mathbf{v}_e^{t+1}))$ with a read-out function $r(\cdot)$.

So the procedure of solving the anomaly detection problem becomes:

1. Given a normal sequence $\mathbf{X}_{tr}$, identify representative patterns on each of the time sequence respectively in multivariate time series.
2. With identified representative patterns events, merge similar events across time series to indicate affinity relation between time series.
3. Build a sequence of bipartite event graph for multivariate time series (with a stride β) based on event matching.
4. Analyze the sequence of bi-partite event graphs and derive a model Ψ to describe the intra- and inter-dependency relations.
5. Given a testing sequence $\mathbf{X}_{te}$, repeat (3), apply the model learned in (4) to predict $\hat{A}$ at time window t^*.
6. Derive anomaly scores based on the predicted $\hat{A}$ and original time series with a proposed readout function $r(\cdot)$.

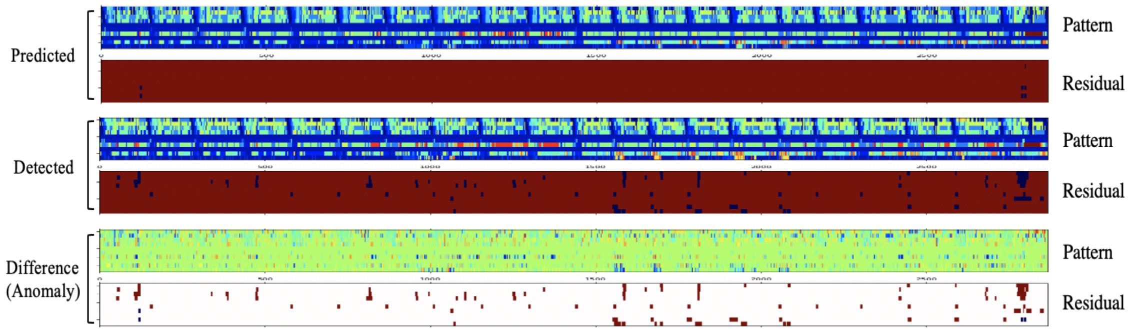

Figure 2: Events are represented by colorful grids in the event bar. In each event bar, X axis corresponding to the time, each row corresponds a time series. One event bar summarizes all the event happened on an individual time series. For each group of event bar, two type event bars corresponding to the events generated by pattern matching and residual computing. The predicted efvent bars are generated by the proposed system via forecasting the next connected event node for each time series. The anomaly score is generated based on the mismatch between the predicted events and the actual events.

5. Event Node Detection And Matching

We defined the event nodes on each time series separately and employed an unsupervised algorithm to identify the events. As we have no prior knowledge about what an anomaly pattern would look like in most real world applications, the system is designed to learn the representative patterns and their correlations from observed data.

Representative segment detection: We adopted Matrix Profile [19], a state-of-the-art unsupervised algorithm to identify representative patterns from time sequences.f Matrix profile is able to identify the top-K repeated patterns on time series with high accuracy and low computation time. We employed a recent implementation of matrix profile called SCRIMP++ [20] on each of the time series in $\mathbf{X}_{tr}$, and the algorithm yields a list of single dimensional representative patterns $\mathbf{p}_{m,k}$ ($m \in \{1 : D\}$, $k \in \{1 : K\}$), each with size τ.

Event generation: many time series share similar representative patterns with each other. It is reasonable to merge similar segment patterns together across different time series to create the event nodes. To measure the similarity between time series, we employed dynamic time wrapping [21], which provides us with a robust distance matrix that is insensitive to small misalignments between two time series segments. After that, we employed the H-DBSCAN [22] to cluster the patterns $\mathbf{p}_{m,k}$ into clusters. We select the centroid of each cluster to be an event, which is a representative segment across all time series within the cluster. Finally, we obtain an event set $\mathbb{E}_\tau$ which contains all events each with length τ.

Event matching: after extracting K events from time series segments, we match each event with the original time series to identify where and when each event is taking place. We employed a highly efficient C language-based Dynamic Time Wrapping algorithm [23] to conduct the event matching, which provides a $(T - \tau)\backslash\beta \times D \times K$ similarity tensor $\mathcal{L}_\tau$ to indicate the similarity of each event to all the sliding window with size τ and stride β in the input.

Event nodes created by pattern matching: for a single time stamp t, given $\mathcal{L}_\tau$ the event graph is created by selecting the best matched events for each time series. The event node is then connected with the corresponding time series node $\mathbf{v}_m^t$ in the event graph $\mathbb{G}_B^t$ with an attribute edge $A(\mathbf{v}_e^t, \mathbf{v}_m^t)$. By finding and linking the best matched event for each time series, we added d number of event nodes into the graph.

Event nodes created by residual error: we noticed that even for the best matched time series, the pattern matching still left with small residual errors. We hereby define two general residual nodes which indicate whether the residual error in a time series is larger than a threshold θ. One residual event node denoted as $\mathbf{v}_{e+}^t$ indicates the residual error is larger than θ, another residual event node denoted as $\mathbf{v}_{e-}^t$ indicates the residual error is equal or smaller than θ. The parameter θ is automatically generated using SPOT [24] algorithm, where we employ the whole training data-set for initialization and testing for on-going adaptation. All time series shared these two residual nodes as shown in Fig. 3.

After generating a sequence of bipartite event graph $\mathbb{G}_B^{(1:t)}$ for time series $\mathbf{X}_{tr}$, a model Ψ is trained on $\mathbb{G}_B^{(1:t)}$ so that given a testing sequence $\mathbf{X}_{te}$, it is able to predict the connectivity of an event graph in any specific time stamp $\dot{t}$ by observing historical sequence observed on $\mathbf{X}_{te}$. In Fig. 2, we visualize the events forecasted/detected on ten time series. The difference between the top and the middle group result in the bottom group where we

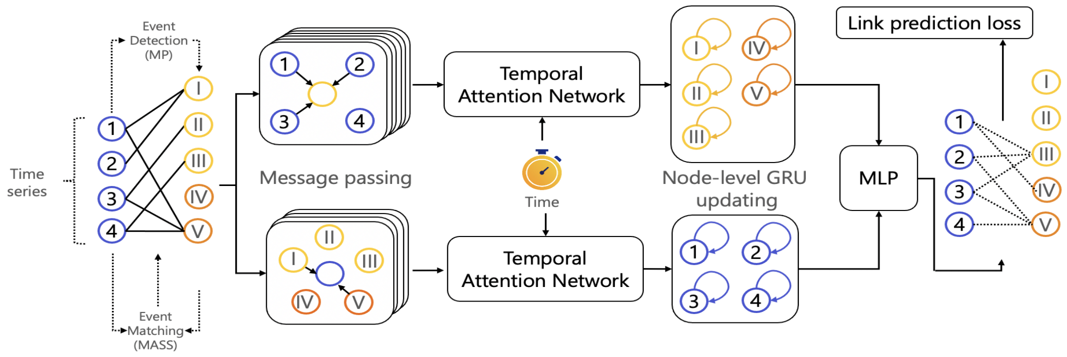

Figure 3: Overview of the proposed approach. The time series nodes are represented by blue circles, two types of event nodes (results from time series matching and residual computation) are represented by yellow and orange circle, respectively. The whole system is trained in an end-to-end manner.

can easily identify the anomaly locations.

5.1. Network Design With Node-wise Memory

For each pair of $\mathbf{v}_m^t$ and $\mathbf{v}_e^t$ in graph $\mathbb{G}_B^{(t)}$, the node features of them are defined as $\boldsymbol{v}_m$ and $\boldsymbol{v}_e$, respectively. We also define $\boldsymbol{\epsilon}_{m,e}$ as the edge features between $\mathbf{v}_m$ and $\mathbf{v}_e$. In order to support the dynamic inter-dependency graph, we avoid explicitly defining a static adjacency matrix to describe the relation between nodes. Instead, we adopt a state-message framework to model the node interactions. For each node $\mathbf{v}_m$ at time stamp t (here we use time series node m as an example; the same rule applies to the event node e), we define a state vector $\boldsymbol{s}_m(t)$ to represent its interaction history with other $\mathbf{v}_e$ nodes before t in a compressed format. By initiating $\boldsymbol{s}_m(0)$ as an all zero vector, the interaction at time t is encoded with a message vector $\boldsymbol{\varrho}_m(t)$:

$$\boldsymbol{\varrho}_m(t) = [\boldsymbol{\epsilon}_{m,e}(t)||\Delta t||\boldsymbol{s}_m(t^-)||\boldsymbol{s}_e(t^-)] \tag{1}$$

where Δt is the time elapse between the previous time stamp t^- and t, symbol $||$ means concatenating operation. After aggregating all the messages from neighbors, the state vector of $\mathbf{v}_m$ is updated as:

$$\boldsymbol{s}_m(t) = mem(agg\{\boldsymbol{\varrho}_m(t_1), ..., \boldsymbol{\varrho}_m(t_b)\}, \boldsymbol{s}_m(t^-)) \tag{2}$$

Here agg($\cdot$) is a general aggregation operation (support learnable parameters). For the sake of simplicity, we only compute the mean of the most recent messages in aggregation. We use mem($\cdot$) to represent a trainable update function (*e.g.*, GRU).

Build upon the updated state vector $\boldsymbol{s}_m(t)$ and $\boldsymbol{s}_m(e)$, a time-aware node embedding can be generated at any time t as following:

$$\boldsymbol{z}_m(t) = \sum_{j \in n_m^k([0,t])} TGA(\boldsymbol{s}_m(t), \boldsymbol{s}_e(t), e_{m,e}, \boldsymbol{v}_m(t), \boldsymbol{v}_e(t)) \tag{3}$$

TGA represents the temporal graph attention module [11], where L graph attention layers compute m's embedding by aggregating information from its L-hop temporal neighbors.

5.2. Optimization and inference

The aforementioned model is trained in a self-supervised fashion. As our goal is to predict the events that might happen in the time sequences in the next time step, which corresponds to the edges linking the time sequence nodes and event nodes. Therefore, we train a simple MLP model with the edge prediction task based on $\boldsymbol{z}_m(t)$.

We use the cross entropy to supervise the forecasting of which original and residual events would happen in a given time stamp t. It is a two hot classification problem where the two activated nodes corresponding to an event node and a residual event node.

Event Forecasting Score: To convert the predicted event edges $\hat{A}$ into an anomaly score, for each time series node $\mathbf{v}_{m,\tau}^t$ with window size τ, the event $\mathbf{v}_{e,\tau}^t$ that has the highest probability connect to $\mathbf{v}_{m,\tau}^t$ is retrieved and its pattern in the original signal space is denoted as $\mathbf{s}_{e,\tau}^t$. We project the event $\mathbf{s}_{e,\tau}^t$'s pattern back to its original signal space, we then compute anomaly score based on the dynamic time wrapping distance as follows:

$$\omega_{1,m}^t = DTW(\mathbf{X}_{m,\tau}^t, \mathbf{s}_{e,\tau}^t) \tag{4}$$

Residual Score: For a positive residual event $\mathbf{v}_{e+}^t$ at time stamp t where the forecasted results is not a positive residual $\hat{\mathbf{v}}_{e-}^t$, we calculate a changing point score to

Table 1
Dataset statistics and hyper-parameters

	SMAP	SMD
Number of Time series nodes	25	38
Number of event nodes	130	64
Time window length (τ)	20	50
Time window stride (β)	5	10

quantify the surprisal level as follows:

$$\omega_{2,m}^{t} = \psi_{NLG}(||\mathbf{X}_{m,\tau}^{t} - \mathbf{X}_{m,\tau}^{t-\tau}||) \tag{5}$$

where ψ_{NLG} is a standard function that maps a scalar into the negative log likelihood, which indicates the sparsity of this changing point in the training data. A frequent changing signal may results in small $\omega_{2,m}^{t}$ after the mapping. The function is learned in a data-driven manner based on the training data of time series m. The final anomaly score at time stamp t is calculated as the following read out function $r(\cdot)$:

$$\omega^{t} = \sum_{m}(\omega_{1,m}^{t} \cdot \omega_{2,m}^{t}) \tag{6}$$

6. Experiment

We performed experiments on two datasets to demonstrate the effectiveness of our model on multivariate anomaly detection. We adopted two public datasets: SMAP (Soil Moisture Active Passive satellite) [2] and SMD (Server Machine Dataset) [4]. We follow the evaluation protocol of [4] and an anomaly is classified a good catch if it is triggered within any subset of a ground truth segment.

6.1. Settings

The proposed bipartite event graph contains $D + K$ number of nodes, where D is the number of time series and K is the number of events (D equals to 25 and 38 in SMAP and SMD datasets, K equal to 130 and 64 on SMAP and SMD datasets). The detailed number of nodes is shown in Table 2. The length (τ) and stride (β) of the time window of each dataset are also displayed in Table 2. For each time series, we set the maximum number of motifs detected to be three, and the minimum cluster size of H-DBSCAN to be three. For the temporal attention model, we set the number of multi-head to be 2. The GRU is employed to model the time encoding. The dimension of node embedding and message vector are both set to 64 respectively. Each model is trained after 10 epochs with the learning rate 0.0001. The node features of both time series and event nodes are randomly initialized, and edge features are the one-hot embedding of the numerical ID of the nodes on both sides of the edge.

Table 2
Compare with current state-of-the-art approaches.

Dataset	Method	Precision	Recall	F1
SMD	DAGMM	59.51	88.82	70.94
	LSTM-VAE	79.22	70.75	78.42
	LSTM-NDT	56.84	64.38	60.37
	OmniAnomaly	83.34	94.49	**88.57**
	MTAD-GAT	-	-	-
	Event2Graph	88.61	83.38	**84.93**
SMAP	DAGMM	58.45	90.58	71.05
	LSTM-VAE	85.51	63.66	72.98
	LSTM-NDT	89.65	88.46	89.05
	OmniAnomaly	74.16	97.76	84.34
	MTAD-GAT	89.06	91.23	**90.13**
	Event2Graph	86.54	94.33	**90.27**

6.2. Compare with state-of-the-art

We compared our solution with multiple baselines on SMAP and SMD datasets: DAGMM [25] - an autoencoder-based anomaly detection model without taking into account of temporal information; LSTM-VAE [26], LSTM-NDT [2], two LSTM-based anomaly detection solutions; and the most recent stochastic VAE-based approaches (*e.g.*, Omni-Anomaly[4]) and graph attention-based method such as MTAD-GAT[5]. We selected these baselines mainly because: (i) they are self-supervised algorithms that do not need any training labels (different from [8]), (ii) they rely on a single scale of time-window (instead of multi-scale [27]) so that the performances are directly comparable.

The results are reported in Table 2. From the results we observed that the proposed Event2Graph achieves competitive performance on SMAP dataset, ranked 2^{nd} on SMD dataset. In SMAP dataset, we observe that some of the data suffers from significant regime change during both training and testing, and our dynamic graph-based solution helps the algorithm adapt to the regime change faster than MTAD-GAT[5] and Omni-Anomaly[4]. We also observe that our algorithm significantly outperforms simple LSTM-based solution (LSTM-VAE and LSTM-NDT), which assumes a complete inter-dependency graph. They did not model the dynamic inter-dependency among time series, while our node-level model explicitly encode the temporal information along with the attention, which helps to reduce the false alarms in anomaly detection. Furthermore, since DAGMM assumes a completely static relationship between time series, the algorithm lacks of the capability to adapt to any temporal evolving pattern.

6.3. Ablation study

The objective of this experiment is to provide detailed analysis over the effectiveness of each proposed module.

By removing each of the critical module of our model, the performances are reported in Table 2.

6.3.1. Effectiveness of temporal graph attention

Through replacing the temporal graph attention with a simple MLP module, the model's F1 score is reduced by 2.42%. It demonstrated that the temporal attention plays an important role in aggregating information from neighborhood nodes.

6.3.2. Effectiveness of event forecasting score ω_1:

We remove the event matching score (Eq. 4) and just use a residual score (Eq. 5) for anomaly detection. We observe that the model performance drop by 5.48%. The experiment indicates the event forecasting module in Event2Graph is essential for accurate anomaly detection. From the experiment, we also observe that TGA plays a critical role in guarantee the quality of the forecasting score ω_1. By removing TGA, the forecasting-based model performance (w/o score ω_2) degrade from 73.81% F1 score to 64.51%.

6.3.3. Effectiveness of residual score ω_2

We remove the residual score (Eq. 5) and only use forecasting score (Eq. 4) to detect anomaly. The model suffers more than 10% performance degradation. This results shows that modeling the residual score is essential to allow the system to identify the anomaly regions that may not well characterized by the event matching scores. We also observe that the TGA may not contribute much to residual event modeling, a ω_2 only model perform competitively well with or without TGA. This phenomenon tells us the residual event is something unexpected and an accurate changing point detection is better than learning simple, repeatable patterns.

6.3.4. Effectiveness of log likelihood function ψ_{NLG}

To demonstrate our assumption, we compare the results with a baseline that using a naive changing point detection score ψ_{NLG}, without adopting any forecasting or event modeling model. Surprisingly, the changing-point only model is able to achieve 78.48% F1-score. Outperforms DAGMM, LSTM-VAE, as well as LSTM-NDT. This results further confirm that it is hard to learn any static pattern by using LSTM or autoencoder for anomaly detection on a challenge dataset, a dynamic model that is able to adapt to the changing of time series patterns is preferred.

Table 3
Ablation study on SMD dataset.

Model	Precision	Recall	F1
Event2Graph	88.61	83.38	84.93
w/o TGA	84.81	82.94	82.51
w/o score ω_1	82.83	80.29	79.45
w/o score ω_2	81.51	71.76	73.81
w/o TGA and w/o ω_1	80.56	81.09	79.47
w/o TGA and w/o ω_2	75.84	64.33	64.51
ψ_{NLG} only	85.89	75.22	78.48

7. Conclusion

We proposed an event-driven bipartite graph solution for multivariate time series anomaly detection. The solution does not assume any inter-dependency on time series, and all the relations are learned in a dynamic, data-driven manner. Our design is based on edge-stream so no adjacency matrix of the graph is required as input. As the system's memory is defined on the node level, our design left plenty space for future extensions such as inductive learning and parallel computation. Our solution achieved very competitive results on two anomaly detection datasets, and we encourage future works to explore further using bipartite event graph for multivariate anomaly detection.

References

[1] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, G. Shroff, Lstm-based encoder-decoder for multi-sensor anomaly detection, CoRR abs/1607.00148 (????).

[2] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Söderström, Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding, KDD' 2018.

[3] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. (1997).

[4] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, D. Pei, Robust anomaly detection for multivariate time series through stochastic recurrent neural network, KDD' 2019.

[5] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, Q. Zhang, Multivariate time-series anomaly detection via graph attention network, ICDM' 2020.

[6] J. Chung, Ç. Gülçehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, CoRR (????).

[7] Z. Cheng, Y. Yang, W. Wang, W. Hu, Y. Zhuang, G. Song, Time2graph: Revisiting time series modeling with dynamic shapelets, AAAI' 2020.

[8] W. Hu, Y. Yang, Z. Cheng, C. Yang, X. Ren, Time-

series event prediction with evolutionary state graph, WSDM '21,.

[9] S. Kumar, X. Zhang, J. Leskovec, Predicting dynamic embedding trajectory in temporal interaction networks, KDD', 2019.

[10] D. Xu, C. Ruan, E. Körpeoglu, S. Kumar, K. Achan, Inductive representation learning on temporal graphs, ICLR' 2020.

[11] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, M. M. Bronstein, Temporal graph networks for deep learning on dynamic graphs, CoRR, abs/2006.10637 (2020).

[12] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, ICLR' 2018.

[13] A. Deng, B. Hooi, Graph neural network-based anomaly detection in multivariate time series, AAAI' 2021.

[14] W. Yu, W. Cheng, C. C. Aggarwal, H. Chen, W. Wang, Link prediction with spatial and temporal consistency in dynamic networks, IJCAI' 2017.

[15] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, W. Wang, Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks, KDD' 2018.

[16] A. Sankar, Y. Wu, L. Gou, W. Zhang, H. Yang, Dysat: Deep neural representation learning on dynamic graphs via self-attention networks, WSDM', 2020.

[17] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, S. Kim, Dynamic network embeddings: From random walks to temporal random walks, IEEE International Conference on Big Data, 2018.

[18] R. Trivedi, M. Farajtabar, P. Biswal, H. Zha, Dyrep: Learning representations over dynamic graphs, ICLR' 2019.

[19] C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, E. J. Keogh, Matrix profile I: all pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets, ICDM' 2016.

[20] Y. Zhu, C. M. Yeh, Z. Zimmerman, K. Kamgar, E. J. Keogh, Matrix profile XI: SCRIMP++: time series motif discovery at interactive speeds, ICDM' 2018.

[21] D. J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series, AAAI' (Workshop) 1994.

[22] L. McInnes, J. Healy, S. Astels, hdbscan: Hierarchical density based clustering, J. Open Source Softw, 2017 (????).

[23] A. Mueen, Y. Zhu, M. Yeh, K. Kamgar, K. Viswanathan, C. Gupta, E. Keogh, The fastest similarity search algorithm for time series subsequences under euclidean distance, 2017.

[24] A. Siffer, P.-A. Fouque, A. Termier, C. Largouet, Anomaly detection in streams with extreme value theory, KDD' 2017.

[25] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, H. Chen, Deep autoencoding gaussian mixture model for unsupervised anomaly detection, ICLR' 2018.

[26] D. Park, Y. Hoshi, C. C. Kemp, A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder, IEEE Robotics and Automation Letters (2018).

[27] L. Shen, Z. Li, J. T. Kwok, Timeseries anomaly detection using temporal hierarchical one-class network, NIPS' 2020.