AnoFusion: Robust Multimodal Failure Detection for Microservice Systems

Chenyu Zhao, Minghua Ma, Zhenyu Zhong, Shenglin Zhang,
Zhiyuan Tan, Xiao Xiong, LuLu Yu, Jiayi Feng, Yongqian Sun, Yuzhi Zhang,
Dan Pei, Qingwei Lin, Dongmei Zhang

CONTENTS

01

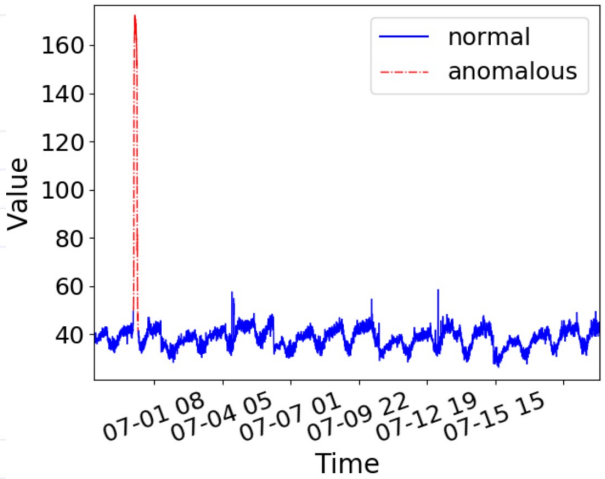
Introduction

Introduction

- Microservice systems are widely used in modern software development.
- They consist of multiple service instances that communicate with each other.
- Failures in one instance can affect the entire system.

Background

- Operators continuously collect three types of monitoring data, including metrics, logs, and traces for proactively detecting instance failures.



Metrics

Logs Templates

ERROR|<*>|bservsice1|db_helper.py ...

INFO | <*> | webservice1 | ...

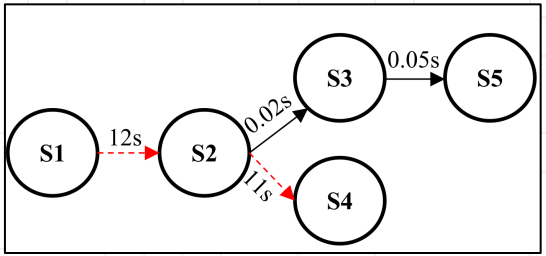
ERROR | <*> | <*> | dbservice1 | ...

IPAddress POST <*> HTTP/1.1 <*> ...

INFO | <*> Deploying application <*>

WARNING | GC <*> | <*> ...

Logs

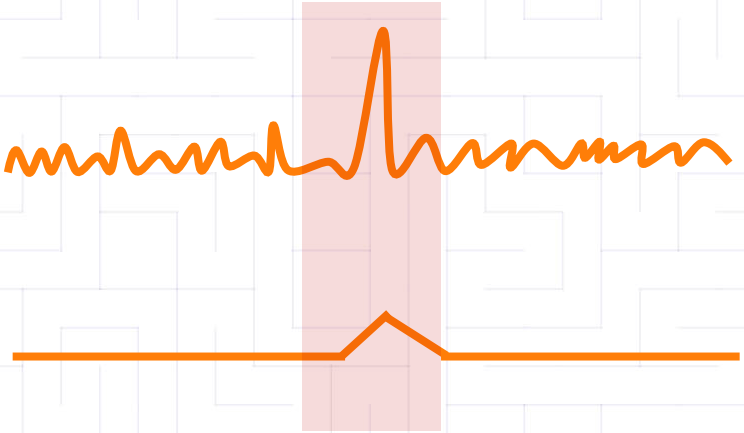


Traces

Motivation

For metrics-based anomaly detection methods:

- Frequent fluctuations can be judged as anomalies.
- Result in a large number of false positives.



Motivation

For logs-based failure detection methods:

- Focus on keywords such as “error”.
- Some failures do not manifest themselves obviously in logs.
- Some anomalous logs do not indicate an instance failure.
- Result in a large number of false positives and false negatives.

Timestamp	Content
0	ERROR $I<*>|S dbservsice1 I dbl
90	INFO $<*>S webservice1 I
180	IPAddress POST $<>$ HTTP/1.1 $<*>$...
270	INFO $\| <*>$ Deploying application $<*>$
360	ERROR I Server $<*>$ is DOWN…
450	INFO $\| <*>\|$ proxy $<*>$ has no server

Motivation

For traces-based failure detection methods:

- Focus on response time.
- A larger response time quickly returning to normal status does not indicate an instance failure.
- Result in a huge number of false positives.

Timestamp	Content
0	S1→S2 Time=0.01s
90	S2→S3 Time=0.32s
180	S3→S4 Time=0.57s
270	S4→S5 Time=11s
360	S5→S6 Time=15s
450	S7→S8 Time=0.78s
540	S8→S9 Time=0.32s

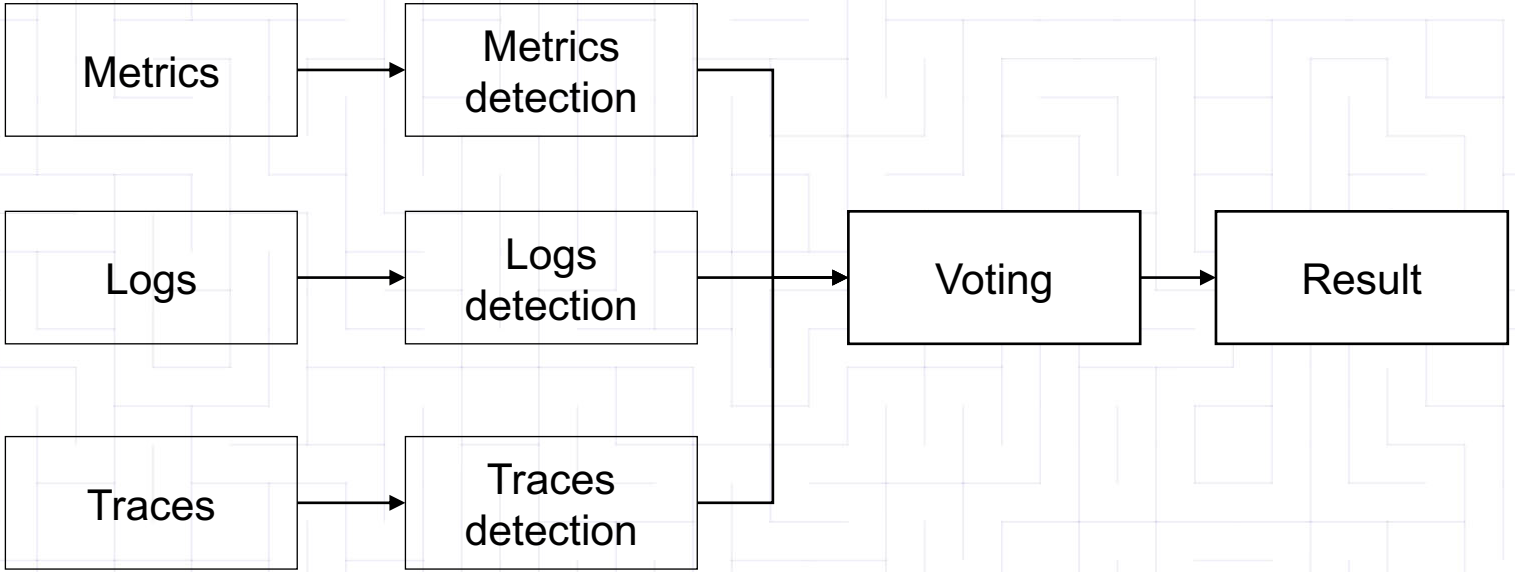
Motivation

- Single-modal data may not be sufficient to detect all types of failures.

Failure Type	Metric	Log	Trace	# Failures
failed of QR code	Mem ↑	–	–	505
system stuck	Mem ↓	–	–	16
login failure	–	ERR	$RT_{S_1 \rightarrow S_2}$=11s	527
file not found	–	–	$RT_{S_2 \rightarrow S_3}$=1.5s	36
access denied	–	ERR	$RT_{S_2 \rightarrow S_4}$=1.1s	15

Motivation

- Require two or more modalities to have anomalies for failure detection.
- It ignores the correlation of the multimodal data.
- Result in many false negatives or false positives.



Motivation

- A failure detection modal.
- Unsupervised method.
- Based on multimodal monitor data.
- Consider the heterogeneity and correlation.
- Handle the dynamically changing of data.

02

Challenge

Challenge1

Modeling the complex correlations among multimodal data.

- When a failure occurs, one, two, or three modalities of data can become anomalous, and they are correlated with each other.
- Neglecting the correlations can degrade the failure detection accuracy.

Challenge2

Dealing with the heterogeneous and dynamically of multimodal data.

- Metrics are usually in the form of multivariate time series.
- Logs are typically semi-structured text.
- Traces consist of spans in a tree structure.
- Integrating such heterogeneous multimodal data is quite challenging.
- An instance’s multimodal data usually changes dynamically over time.

03

Contribution

Contribution1

Modeling the complex correlations among multimodal data.

Apply Graph Transformer Network (GTN).

Contribution2

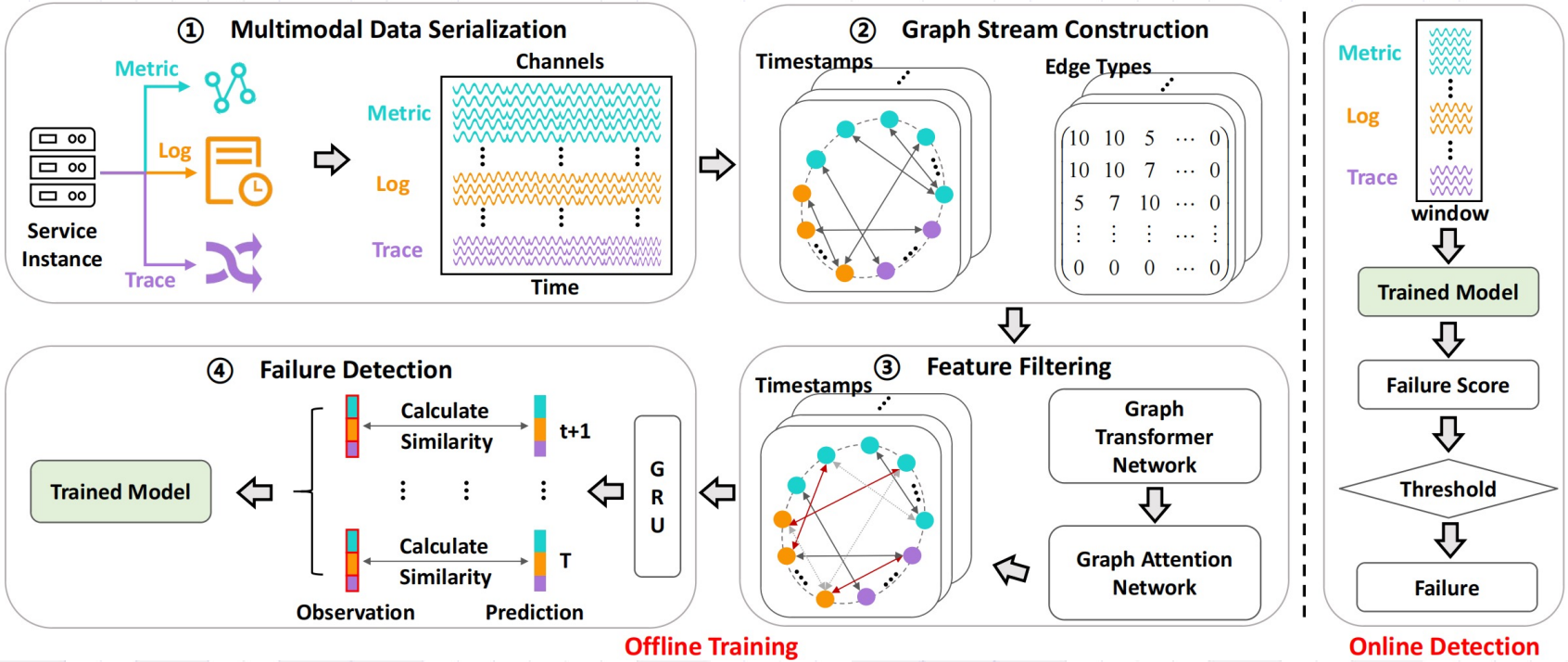
Dealing with the heterogeneous and dynamically changing multimodal data.

Serialize the data of each modality and adopt GAT and a GRU.

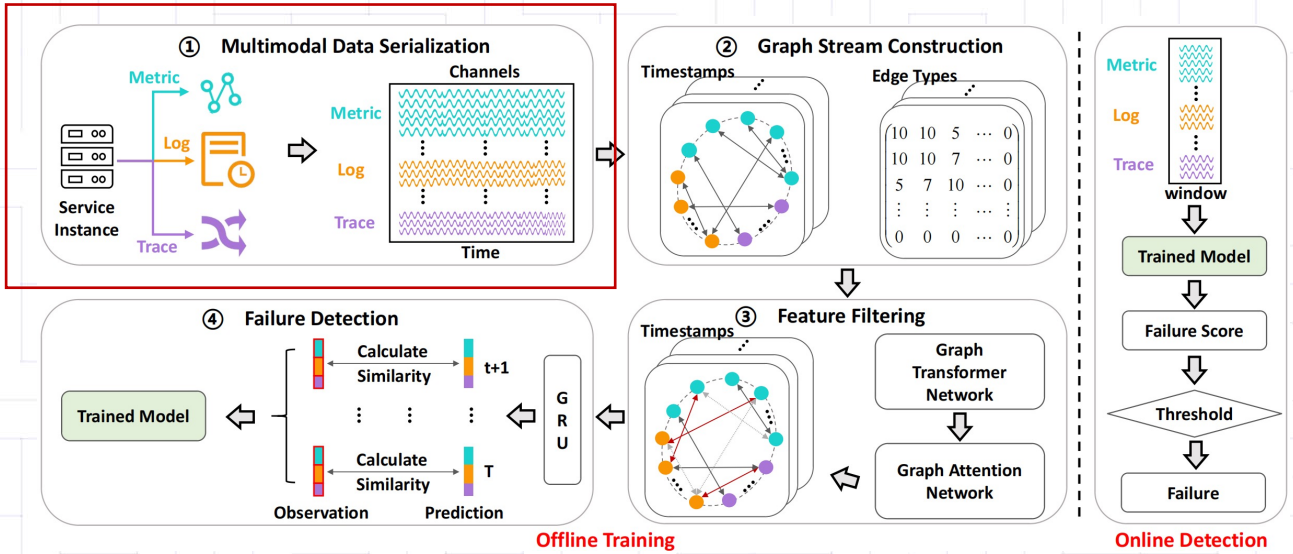
04

Architecture

Architecture



Architecture



- Metric data: Regular preprocessing steps.
- Log data: By clustering and sliding windows.
- Trace data: Response time and status code.

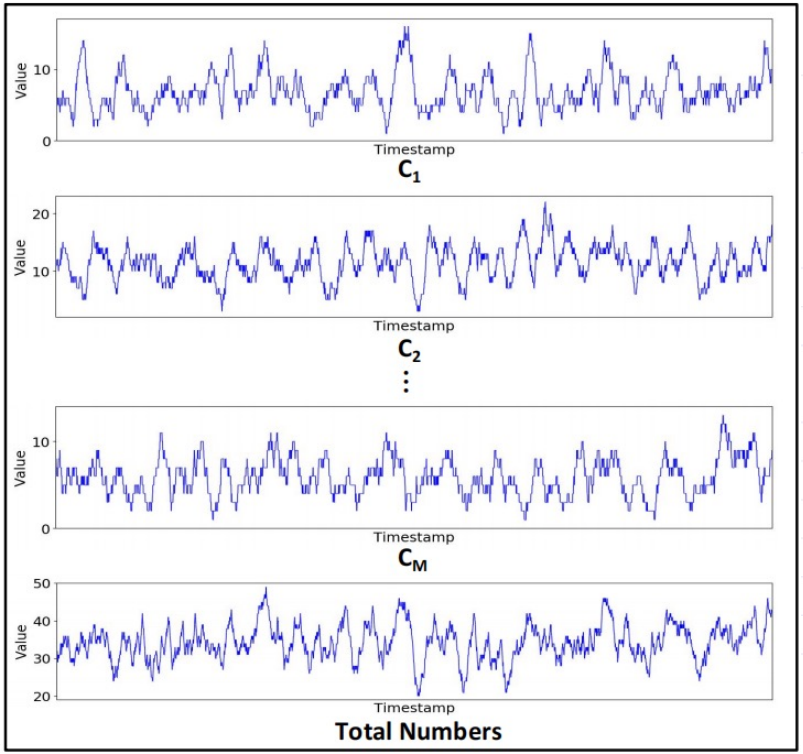
Architecture

T_1: info <*> write success
T_2: INFO | <*> | <*> | dbservice2 | <*> | the list of all available services are redisservice1: http:/<*>, redisservice2: http:/<*>",1195
T_3: <*>-<*>-<*> <*>:<*>:<*>,<*> | INFO | <*> | dbservice2 | db_helper.py -> db_login_methods -> <*> | <*> | <*> token generate success <*>"
…

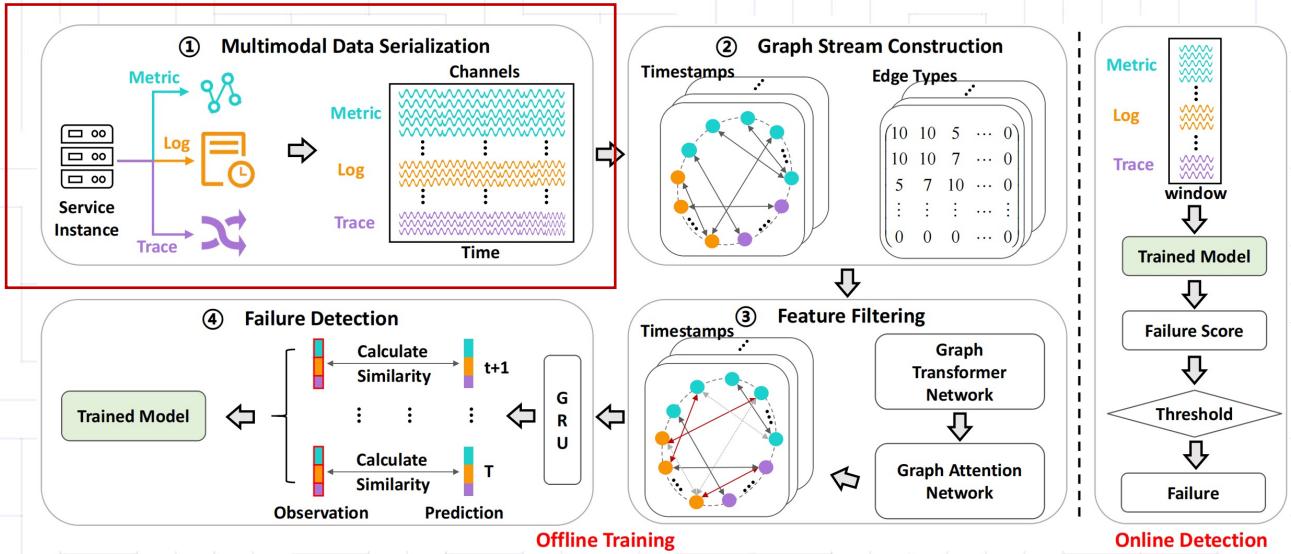
① Clustering (DBSCAN)

C_1: info <*> success
C_2: <*> | dbservice2 | db_helper.py -> db_login_methods -> <*>
C_3: <*>-<*>-<*> <*>:<*>:<*>,<*> | INFO | <*>
…

② Serialization

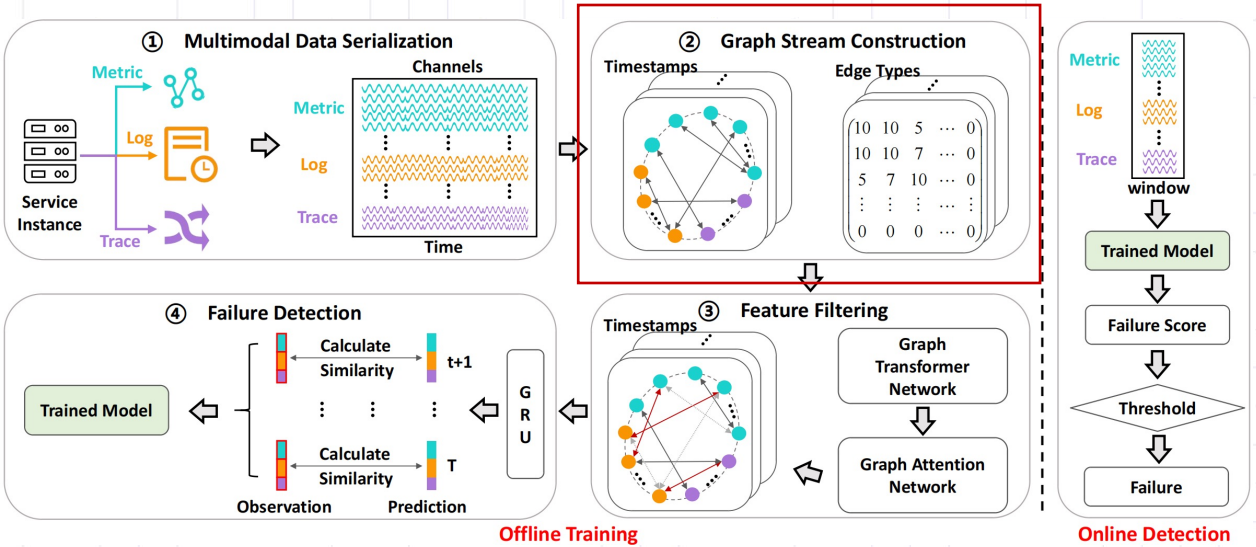


Architecture



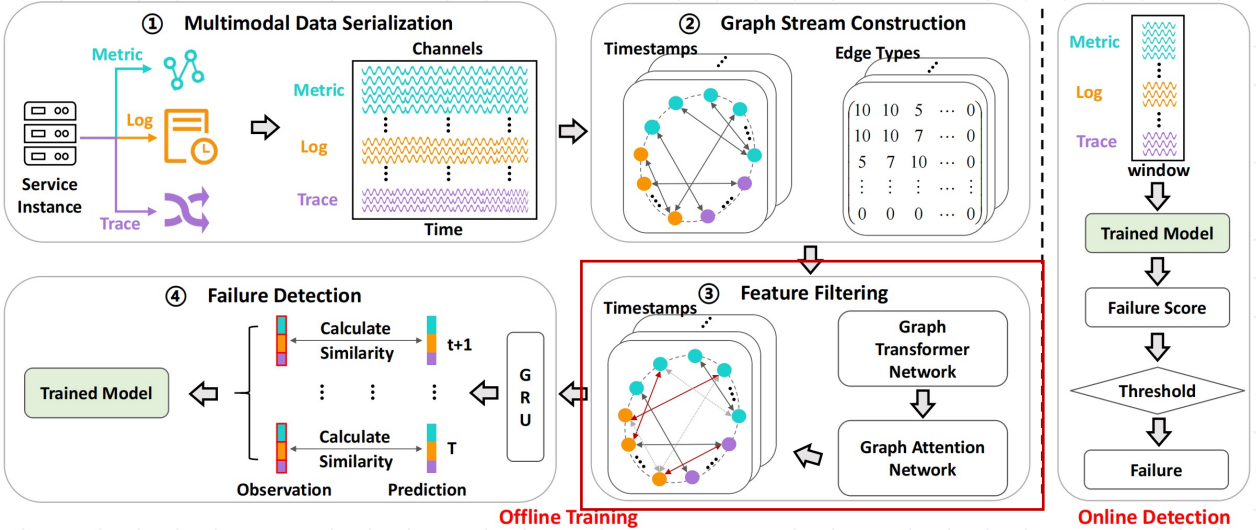
- Metric data: Regular preprocessing steps.
- Log data: By clustering and sliding windows.
- Trace data: Response time and status code.

Architecture



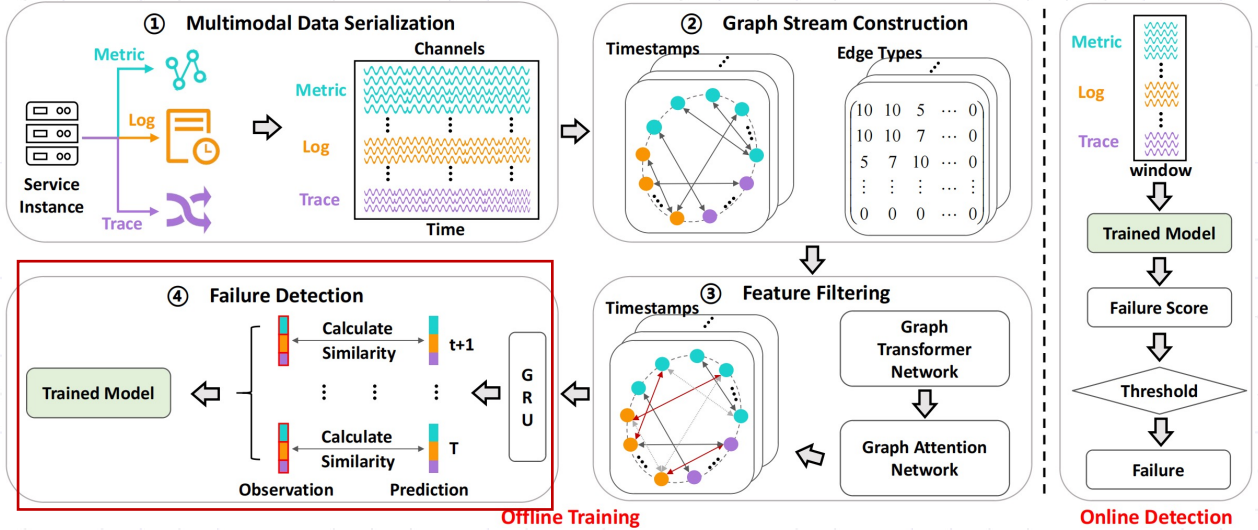
Construct a heterogeneous graph for each time using the extracted data channels.

Architecture



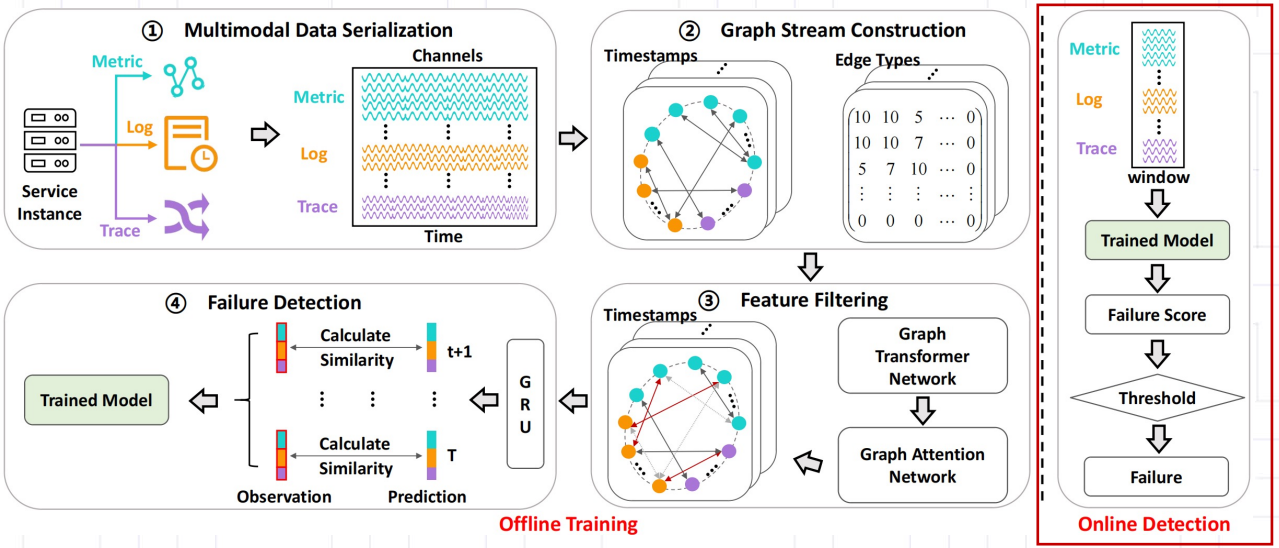
- GTN is used to capture the correlation among different data modalities.
- The GAT is used to identify different patterns and achieving feature filtering.

Architecture



- GRU is applied to temporal sequences to predict the values at the next moment based on the previous inputs.

Architecture



- Serialize the data using its previous historical observations.
- Construct the graph stream
- Get a prediction vector and calculate the failure score.

05

Evaluation

Datasets

Dataset 1

Number of Microservices	Number of Instances	Failures (%)	Modality	#
5	10	4.908	Metric	734,165
			Log	87,974,577
			Trace	28,681,438

Dataset 2

Number of Microservices	Number of Instances	Failures (%)	Modality	#
14	28	1.243	Metric	3,122,168
			Log	14,894,069
			Trace	9,473,763

Effectiveness

Approach	Modality			D1			D2		
	Metric	Log	Trace	Precision	Recall	F_1-score	Precision	Recall	F_1-score
JumpStarter [25]	✓			0.466	0.785	0.584	0.533	0.413	0.465
USAD [1]	✓			0.459	0.825	0.590	0.837	0.341	0.484
LogAnomaly [27]		✓		0.486	0.957	0.644	0.126	0.344	0.184
Deeplog [5]		✓		0.506	0.812	0.623	0.105	0.275	0.151
TraceAnomaly [21]			✓	0.550	0.548	0.549	0.521	0.699	0.597
SCWarn [46]	✓	✓		0.547	0.425	0.447	0.633	0.891	0.734
JLT	✓	✓	✓	0.461	0.940	0.618	0.800	0.344	0.481
AnoFusion	✓	✓	✓	**0.795**	**0.945**	**0.857**	**0.863**	**0.991**	**0.922**

Conclusion

- We propose AnoFusion, one of the first studies using multimodal data, i.e., metrics, logs, and traces, to detect failures of instances in microservice systems robustly.
- We apply AnoFusion on two microservice systems, which proves that it significantly improves the F1-score for failure detection.
- We believe that the solution of applying multimodal data for failure detection will benefit more areas beyond microservice systems.

Thank you!

Q&A