



---

# Heterogeneous Anomaly Detection for Software Systems via Semi-supervised Cross-modal Attention

---

**Cheryl Lee\***, Tianyi Yang\*, Zhuangbin Chen\*, Yuxin Su<sup>†</sup>, Yongqiang Yang<sup>‡</sup>, and Michael R. Lyu\*

\*The Chinese University of Hong Kong, <sup>†</sup>Sun Yat-sen University, <sup>‡</sup>Huawei Cloud

May, 2023

# Table of Contents

01

**INTRODUCTION**

02

**MOTIVATION**

03

**METHODOLOGY**

04

**EVALUATION**



01

# INTRODUCTION

Background, Preliminary...



# Background



## Anomaly detection is essential

**Twitter back after two-hour outage affected tweets**

© 1 March

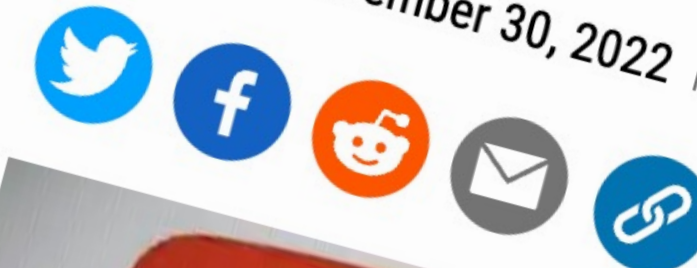


**Facebook Lost About \$65 Million During Hours-Long Outage**

Abram Brown Former Staff

Listen to article 2 minutes

Oct 5,



**YouTube App Down on iOS Devices**

Users reported a YouTube outage on Wednesday, complaining the app crashed logging on.

By Nikki Main

Published November 30, 2022 | Alerts

**Amazon's one hour of downtime on Prime Day may have cost it up to \$100 million in lost sales**

Sean Wolfe Jul 19, 2018, 10:53 PM





# Background



## Single-source data may be insufficient

The screenshot displays the 'JVM GC Logs Demo' application interface. The top navigation bar includes 'Logs > JVM GC Logs Demo > Overview', a 'Sign up' button, and a 'Last hour' filter. A search bar is present with the text 'Search your logs'. The main content area is divided into several sections:

- TIP:** You should see most time spent in Young GC (more efficient). If not, try increasing young gen size or MaxGCPauseMillis.
- Young GC:** 8.4kms
- Old GC Total:** 29.8kms
- TIP:** Too much time in GC (e.g. more than 10%) is a sign that changes to GC settings and/or heap size or application code should help.
- Total Garbage Collection Time:** A bar chart showing GC time intervals from 06:35 to 07:33. The y-axis ranges from 0 to 3.50 s.
- CPU Timing:** A line chart showing System (green), User (orange), and Real (light orange) CPU times from 06:35 to 07:32. The y-axis ranges from 0 to 1.40 s.
- TIP:** Real is wall clock time. User is actual work, counted per core (should be >Real). System is work outside the JVM. If it's not minimal, it's a red flag (e.g. ...).
- TIP (for G1):** The before-after difference gives an idea on the amount of headroom. Too little means GC pressure: either GC is falling behind or there's not enough heap. Too much (e.g. before > 3 x after) means you could do with less heap.
- G1 Used Before:** A bar chart showing memory usage from 06:35 to 07:35, with values between 0 and 700.
- G1 Used After GC:** A bar chart showing memory usage after GC from 06:35 to 07:35, with a peak around 07:10 reaching approximately 3.50k.

On the right side, there is a 'Type to filter fields' input and a list of filterable fields including @timestamp, @timestamp\_received, Abstract\_VM\_Version\_jdk\_debug\_le..., Abstract\_VM\_Version\_vm\_release, AlwaysPreTouch, ShenandoahHeapRegion\_region\_size..., after\_archive\_length, after\_eden\_length, after\_humongous\_length, after\_old\_length, after\_survivor\_length, age, before\_archive\_length, before\_eden\_length, before\_humongous\_length, before\_old\_length, before\_survivor\_length, and capacity.



# Background



**Anomaly detection is essential**



**Single-source data may be insufficient**



**Combining multi-source data may be effective**



# PRELIMINARY

## Logs

```
17/06/09 20:10:48 INFO executor.Executor: Finished task 0.0 in stage 0.0 (TID 0). 2703 bytes result sent to driver
17/06/09 20:10:52 INFO executor.CoarseGrainedExecutorBackend: Got assigned task 42
17/06/09 20:10:52 INFO executor.Executor: Running task 0.0 in stage 1.0 (TID 42)
17/06/09 20:10:52 INFO executor.CoarseGrainedExecutorBackend: Got assigned task 56
17/06/09 20:10:52 INFO executor.Executor: Running task 1.0 in stage 1.0 (TID 56)
```

Log Sequence

Parsing

Log Message

Timestamp	Level	Component	Log Event
17/06/09 20:10:48	INFO	executor.Executor	Finished task * in stage * (TID *). * bytes result sent to driver.
17/06/09 20:10:52	INFO	executor.CoarseGrainedExecutorBackend	Got assigned task *
17/06/09 20:10:53	INFO	executor.Executor	Running task * in stage * (TID *)
17/06/09 20:10:54	INFO	executor.CoarseGrainedExecutorBackend	Got assigned task *
17/06/09 20:10:55	INFO	executor.Executor	Running task * in stage * (TID *)

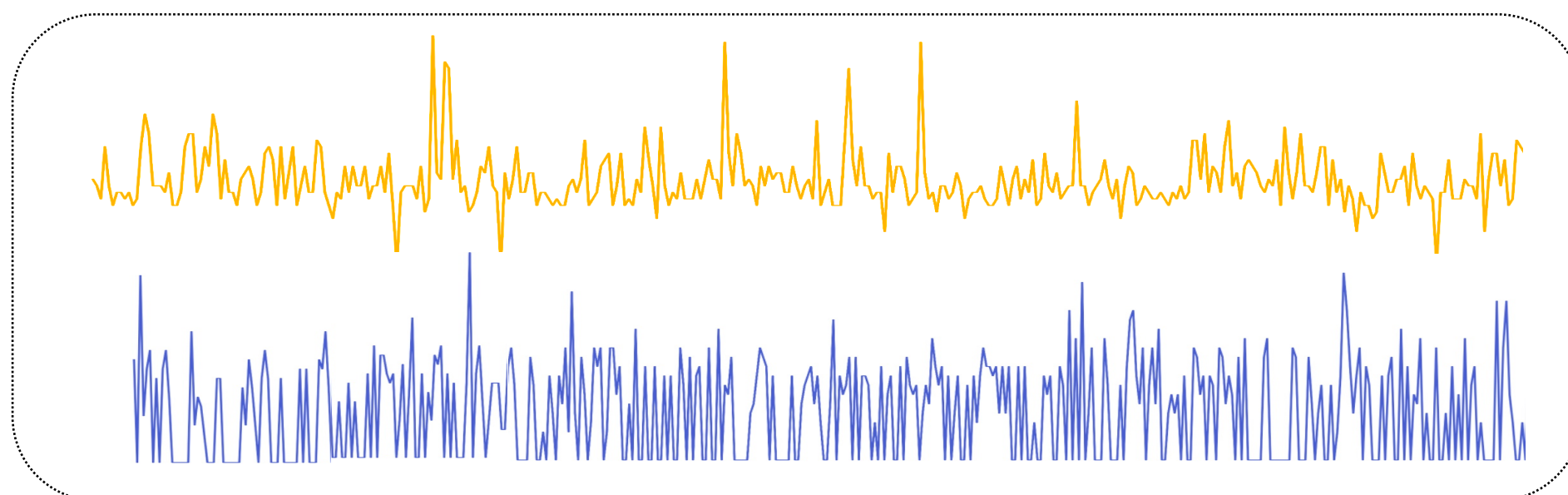


# PRELIMINARY

## Log events

INFO util.SignalUtils: Registered signal  
WARN netlib.BLAS: Failed to load implementation  
INFO storage.BlockManager: Removing RDD 36  
INFO util.Utils: Successfully started service  
INFO storage.BlockManager: Removing RDD 18

## Metrics





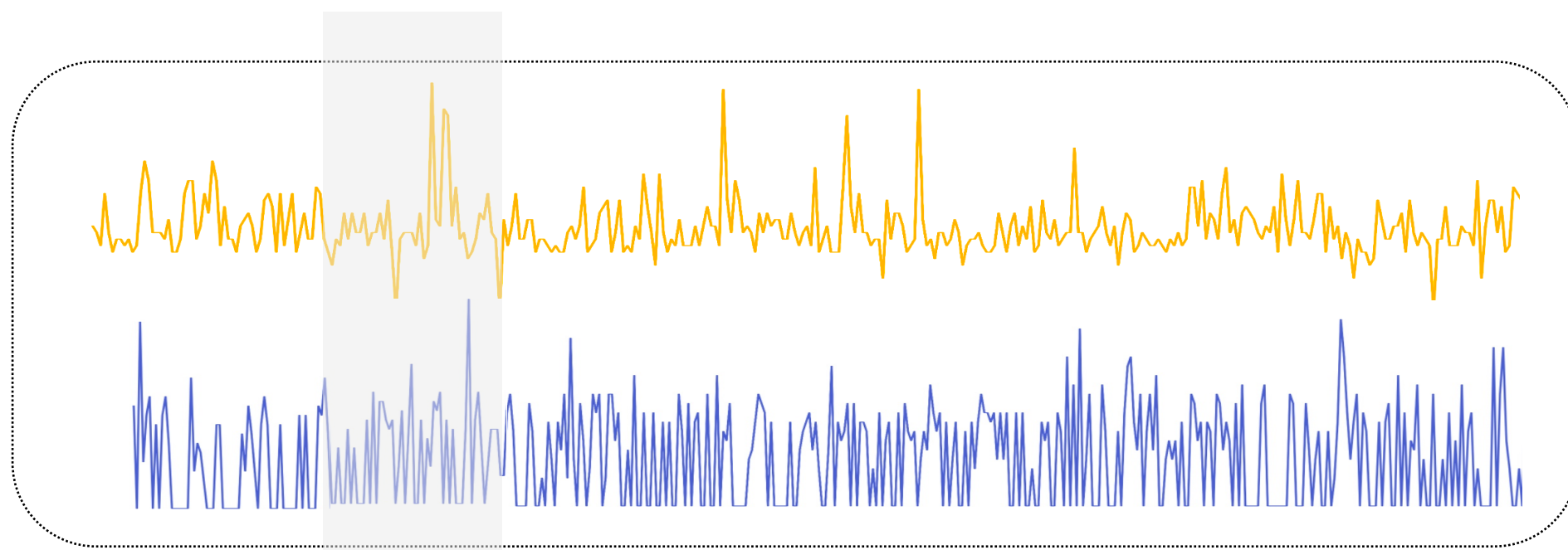


# PRELIMINARY

## Log events

INFO util.SignalUtils: Registered signal  
WARN netlib.BLAS: Failed to load implementation  
INFO storage.BlockManager: Removing RDD 36  
INFO util.Utils: Successfully started service  
INFO storage.BlockManager: Removing RDD 18

## Metrics

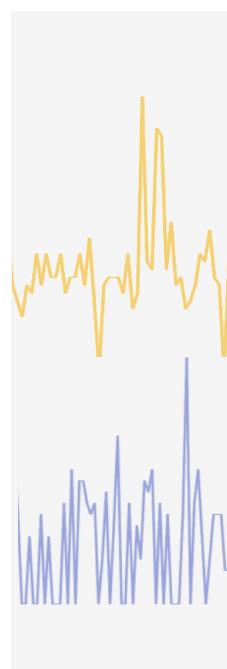




# PRELIMINARY

```
WARN netlib.BLAS: Failed to load implementation  
INFO storage.BlockManager: Removing RDD 36  
INFO util.Utils: Successfully started service  
INFO storage.BlockManager: Removing RDD 18
```

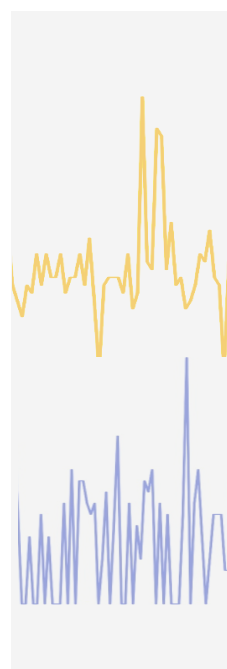
**A chunk**





# PRELIMINARY

```
WARN netlib.BLAS: Failed to load implementation  
INFO storage.BlockManager: Removing RDD 36  
INFO util.Utils: Successfully started service  
INFO storage.BlockManager: Removing RDD 18
```



**A chunk**



?



?



02

## MOTIVATION

Anomaly Characteristics, Case Studies

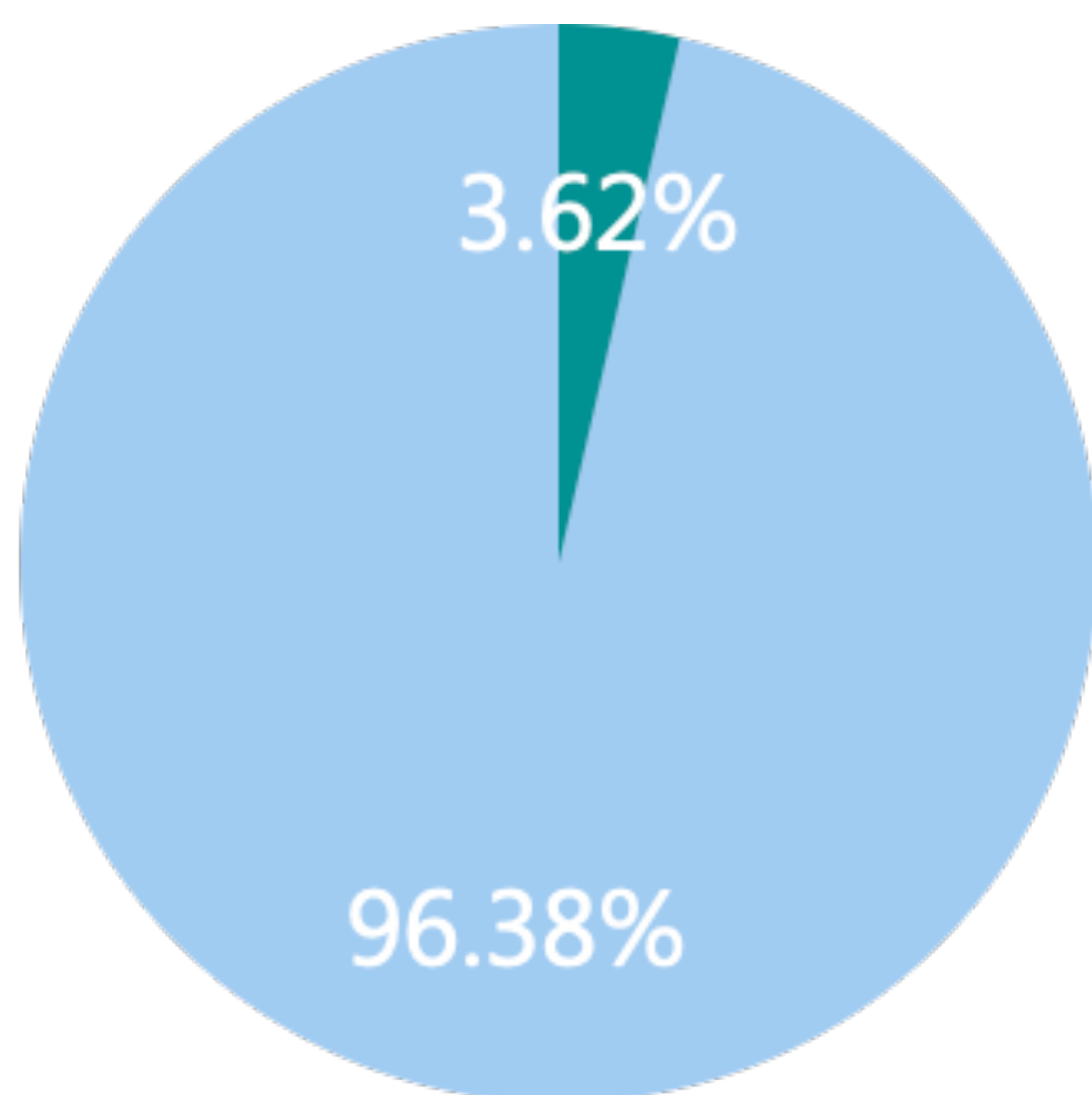


# How do logs manifest system anomalies?



## Finding 1

Logs sometimes cannot record fine-grained information and therefore, are not susceptible enough to manifest all system anomalies.



● Log's contribution

Only 3.62% of positively labeled chunks are anomalous from the log's perspective.

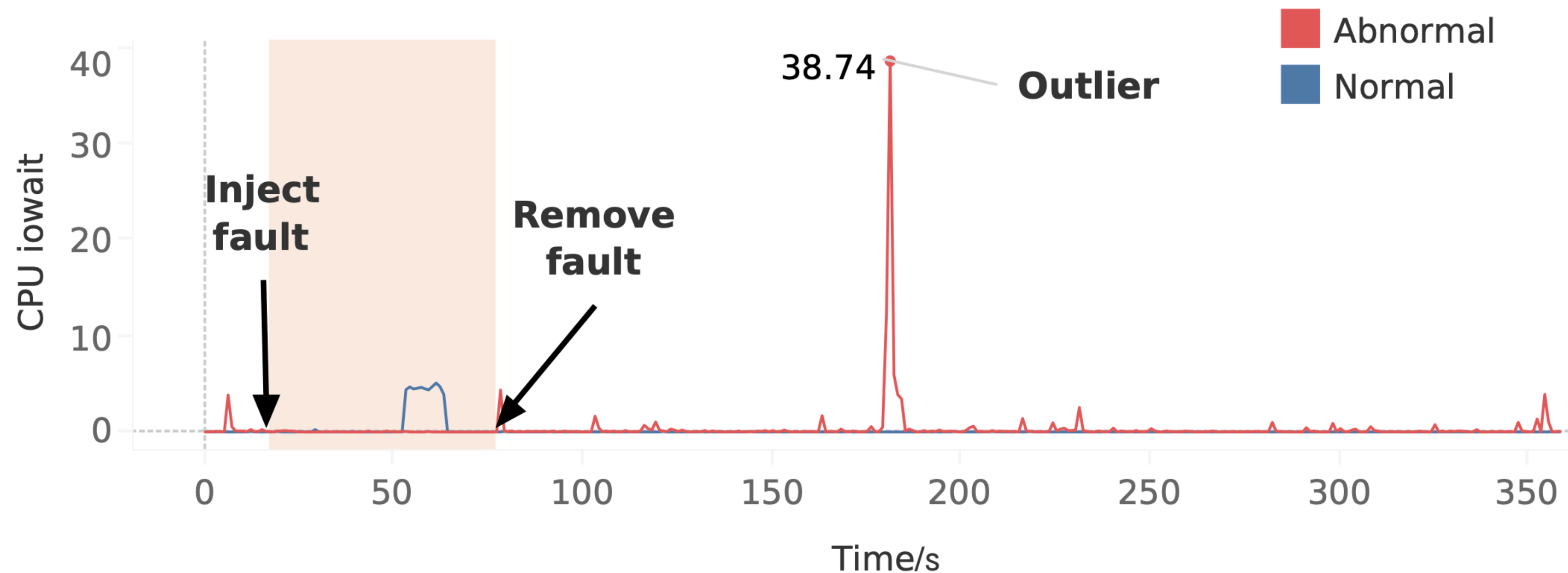


# How do metrics manifest system anomalies?



## Finding 2

Metrics are insufficient sometimes. Their over-sensitivity may cause false alarms on uncommon yet acceptable fluctuations.



“CPU iowait” generates a rare heartbeat spike even in the fault-free period.



# How do logs & metrics manifest anomalies?



## Finding 3

Metrics and logs can both respond to anomalies, but neither is sufficient. They have collaborative and complementary relationships in reflecting anomalies.

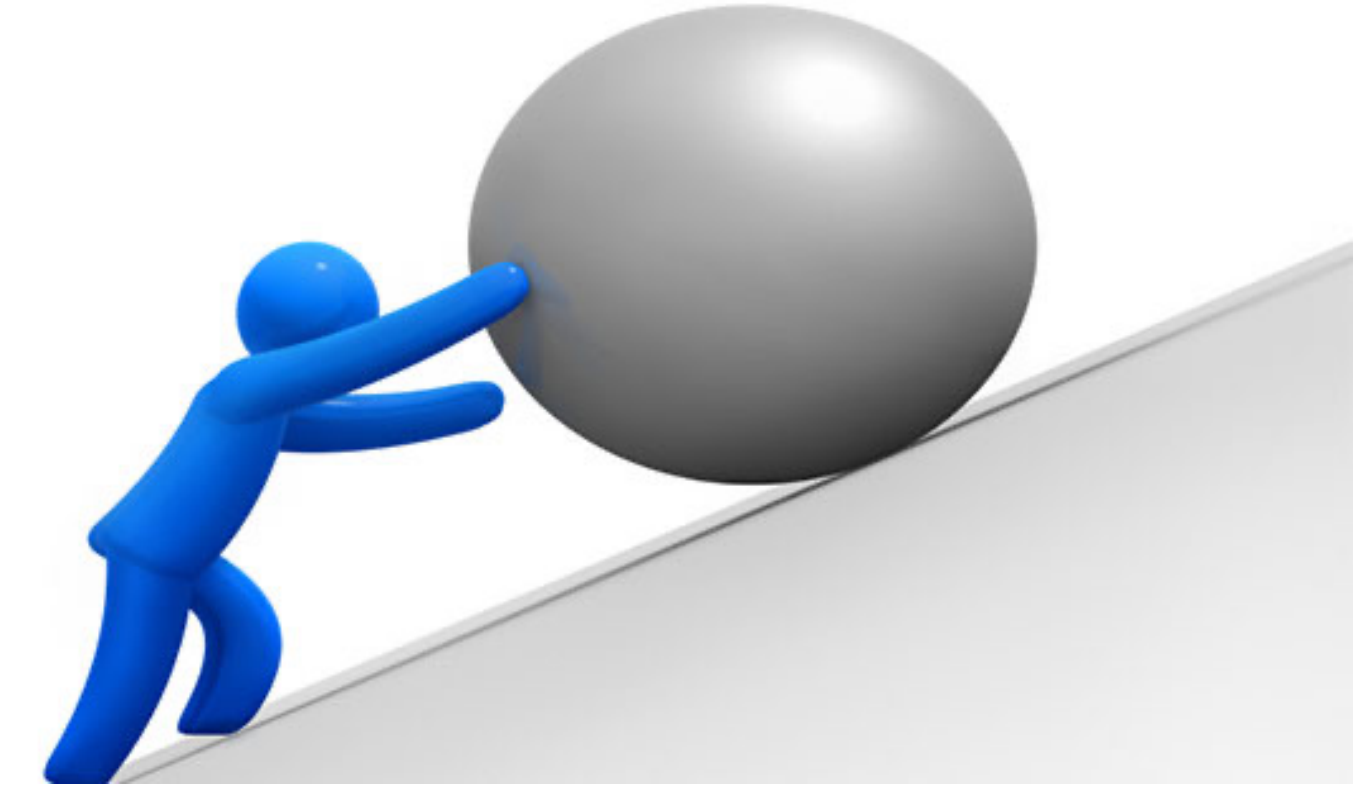
TABLE I: Typical faults and the corresponding anomalous manifestations of logs and metrics

Faults	Anomalies in logs	Anomalies in metrics
Memory hog	Warnings (reaches the memory limit)	Memory-related metrics rise steeply
Virtual memory hog	Errors (reporter thread fails)	CPU and memory-related metrics jitter
I/O hog	Warnings (slow ReadProcessor)	I/O-related metrics rise steeply
Network delay	Warnings (executor heartbeat timeout)	Network-related metrics suddenly drop
Connection flash	Nothing ( <b>silent</b> )	Network-related metrics suddenly drop and quickly restore
Datanode killed	Errors (excluding datanode)	Related metrics plummet to zero ( <b>silent</b> )
Secondary namenode killed	Errors (failed to connect to <IP>)	Related metrics plummet to zero ( <b>silent</b> )

# Challenges

## Complex intra-modal information:

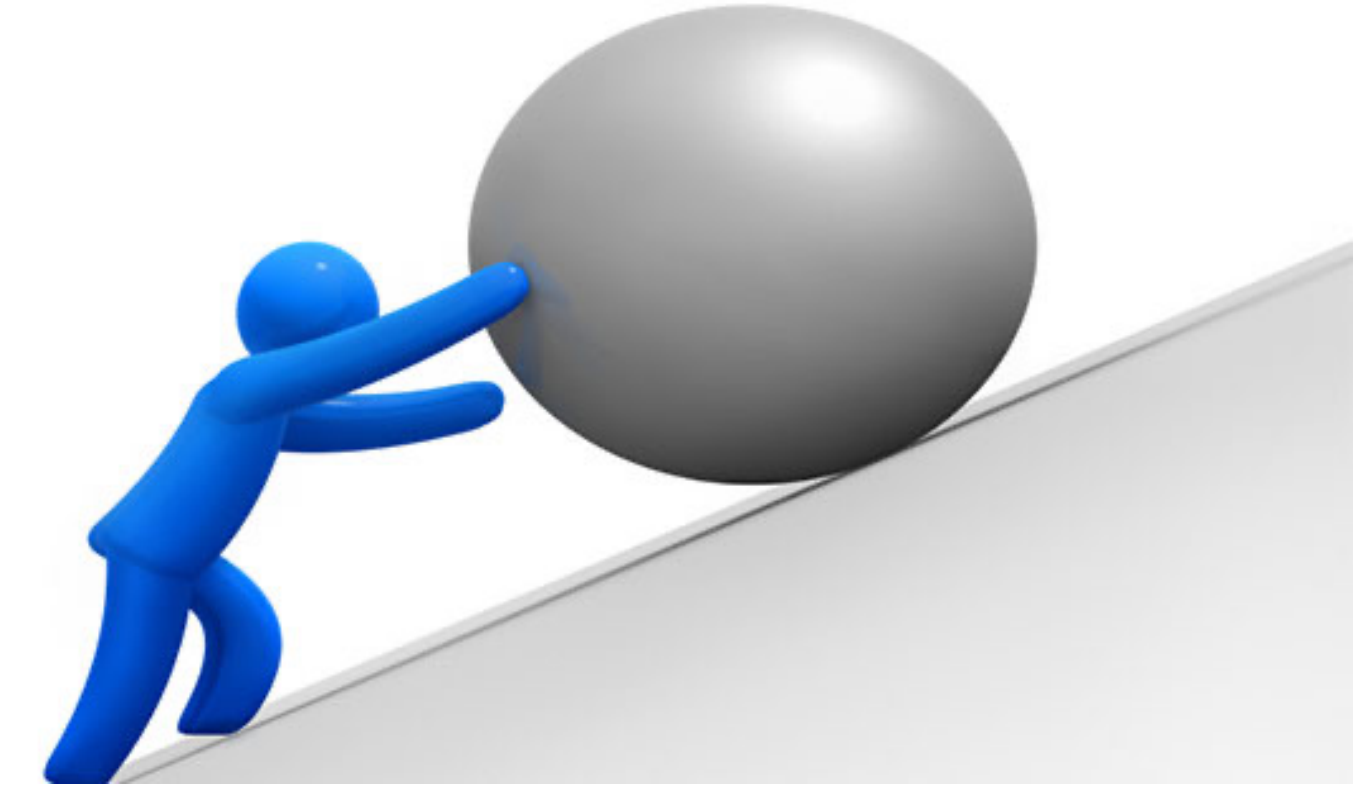
- Log semantics and sequential dependencies.
- Metrics' diverse aspects.







# Challenges



## Complex intra-modal information:

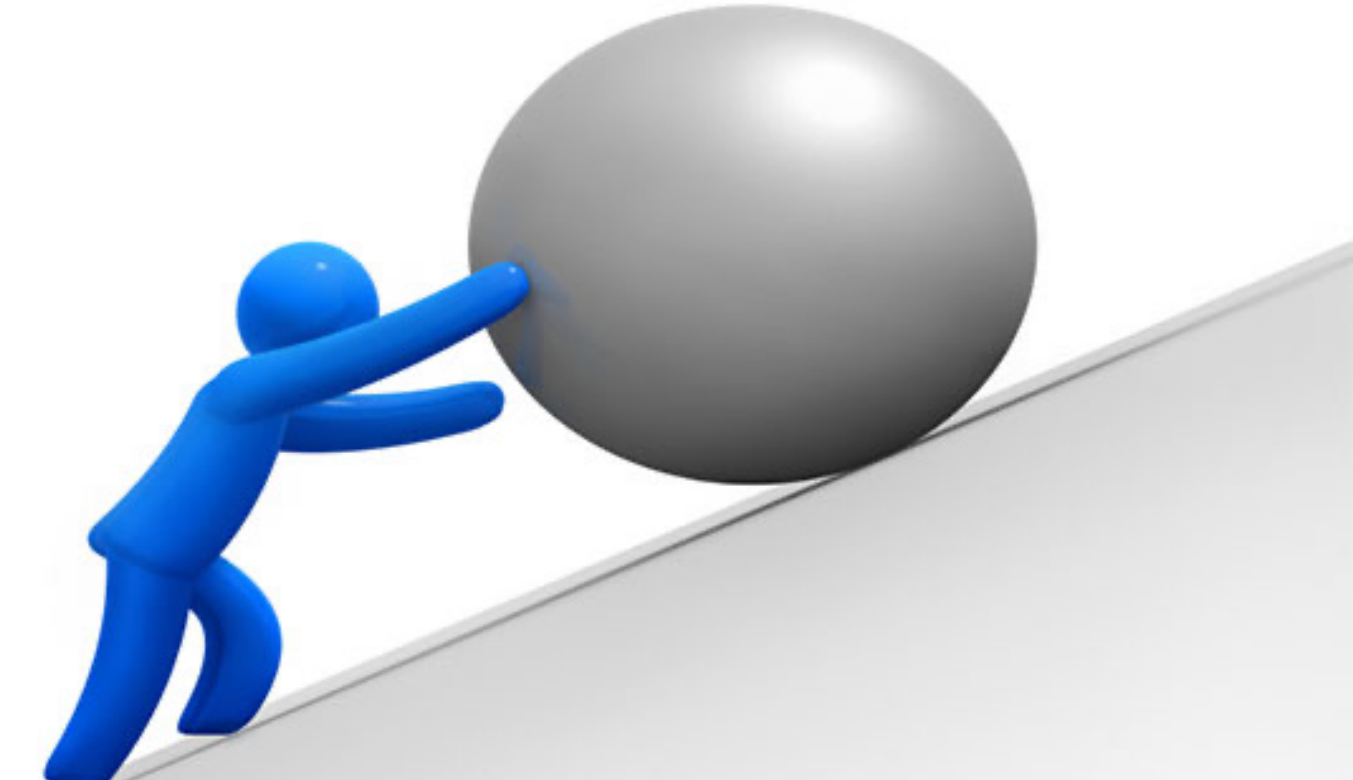
- Log semantics and sequential dependencies.
- Metrics' diverse aspects.

## Significant inter-modal gap:

- Logs and metrics are in different forms.
- Different degrees of anomaly affectedness.



# Challenges



## **Complex intra-modal information:**

- Log semantics and sequential dependencies.
- Metrics' diverse aspects.

## **Significant inter-modal gap:**

- Logs and metrics are in different forms.
- Different degrees of anomaly affectedness.

## **Trade-off between cost and accuracy:**

- Supervised learning is accurate but costly.
- Unsupervised learning ignores human oversight.



# Our Solution

## Complex intra-modal information:

- Log semantics and sequential dependencies.
- Metrics' diverse aspects.



Properly modeling  
each modality

## Significant inter-modal gap:

- Logs and metrics are in different forms.
- Different degrees of anomaly affectedness.

## Trade-off between cost and accuracy:

- Supervised learning is accurate but costly.
- Unsupervised learning ignores human oversight.



# Our Solution

## Complex intra-modal information:

- Log semantics and sequential dependencies.
- Metrics' diverse aspects.



Properly modeling  
each modality

## Significant inter-modal gap:

- Logs and metrics are in different forms.
- Different degrees of anomaly affectedness.



Cross-modal  
attention

## Trade-off between cost and accuracy:

- Supervised learning is accurate but costly.
- Unsupervised learning ignores human oversight.



# Our Solution

## Complex intra-modal information:

- Log semantics and sequential dependencies.
- Metrics' diverse aspects.



Properly modeling  
each modality

## Significant inter-modal gap:

- Logs and metrics are in different forms.
- Different degrees of anomaly affectedness.



Cross-modal  
attention

## Trade-off between cost and accuracy:

- Supervised learning is accurate but costly.
- Unsupervised learning ignores human oversight.



Semi-supervised



03

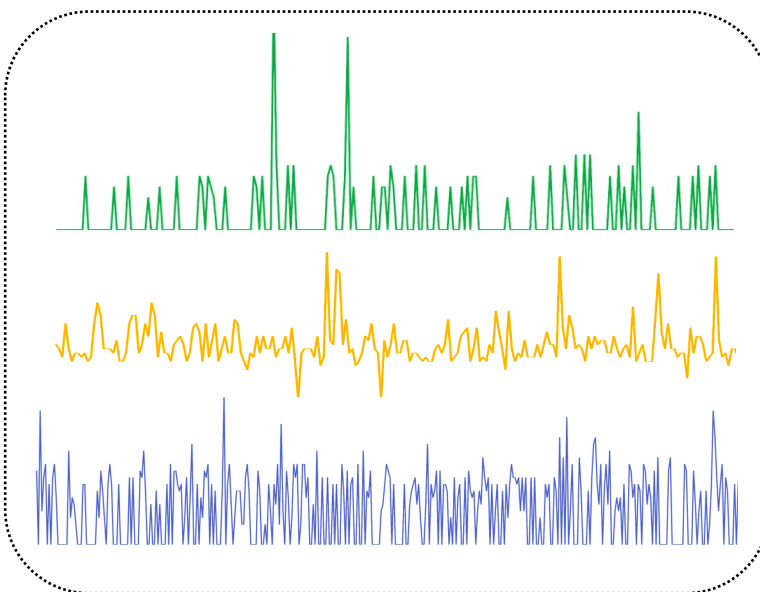
# METHODOLOGY

Modal-wise Modeling, Cross-modal Attention



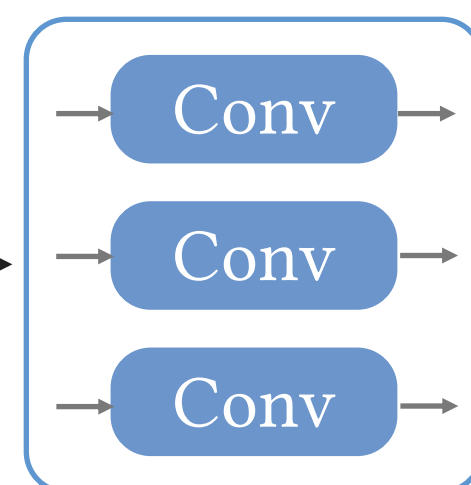
# Overview

## Multivariate Metrics

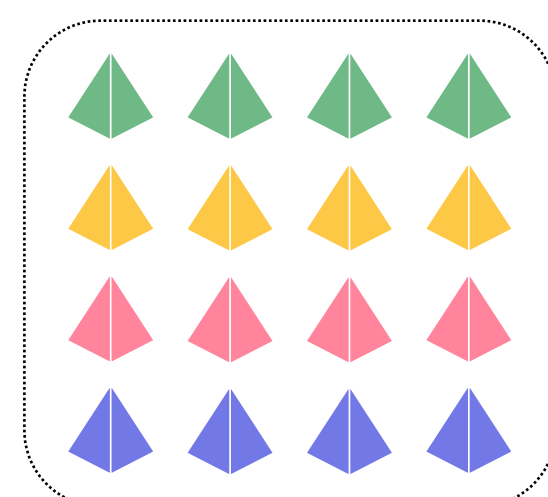


Grouping

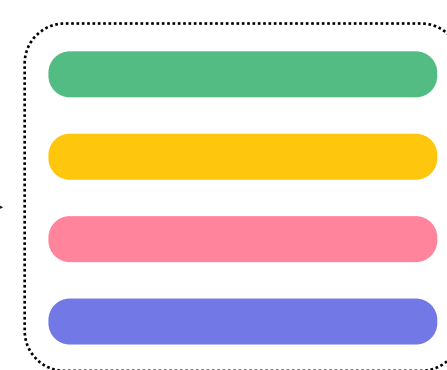
Intra-aspect Encoder



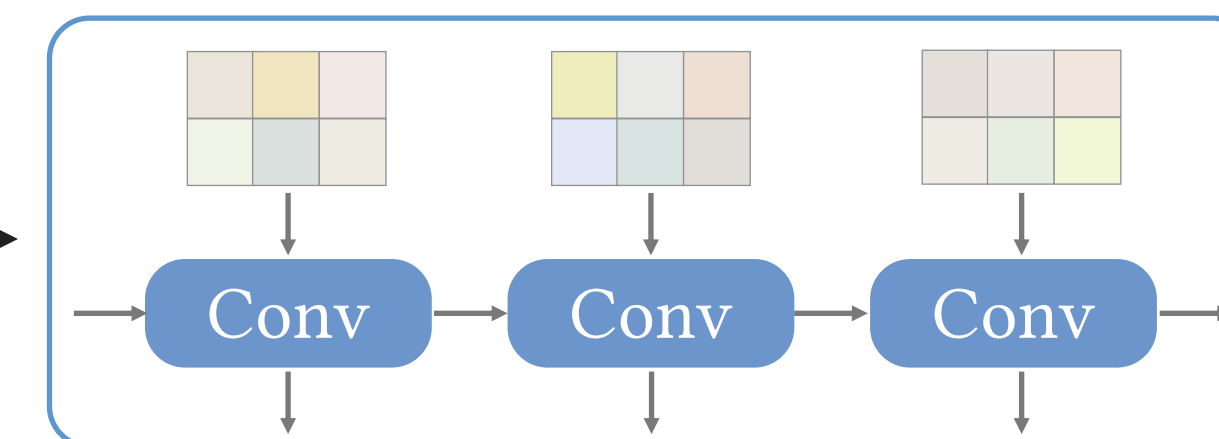
## Metric Modeling



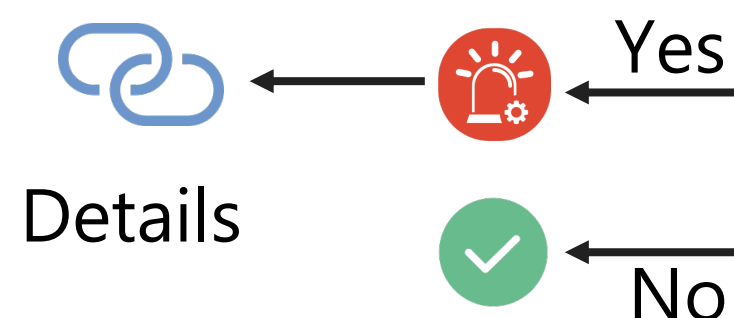
Max Pooling



Inter-aspect Encoder



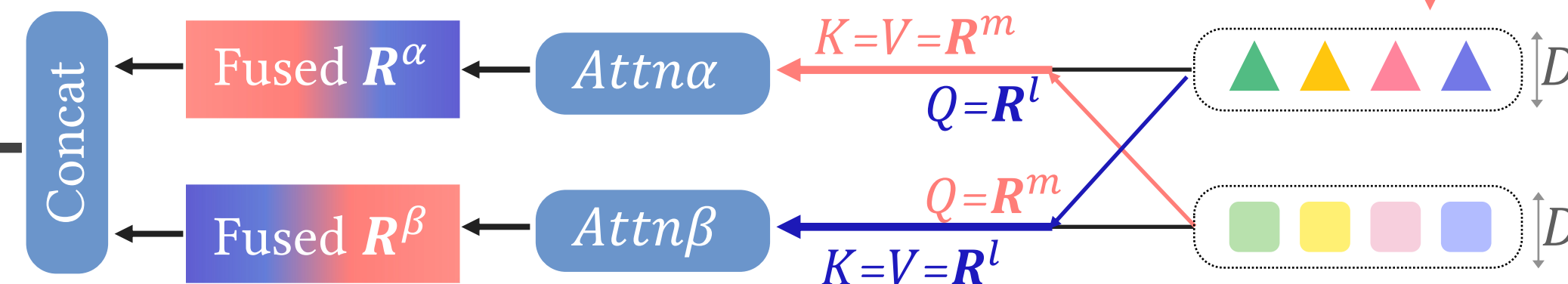
## Detection



FC

Global  $R^g$   $D$

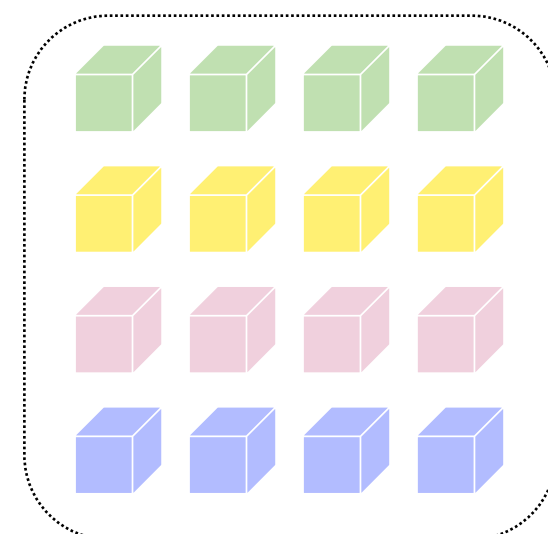
## Heterogeneous Fusion



## Log Modeling

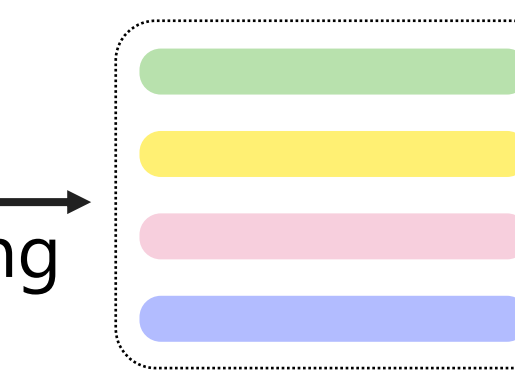
INFO util.SignalUtils: Registered signal  
WARN netlib.BLAS: Failed to load implementation  
INFO storage.BlockManager: Removing RDD 36  
INFO util.Utils: Successfully started service  
INFO storage.BlockManager: Removing RDD 18  
.....

Parsing  
FastText

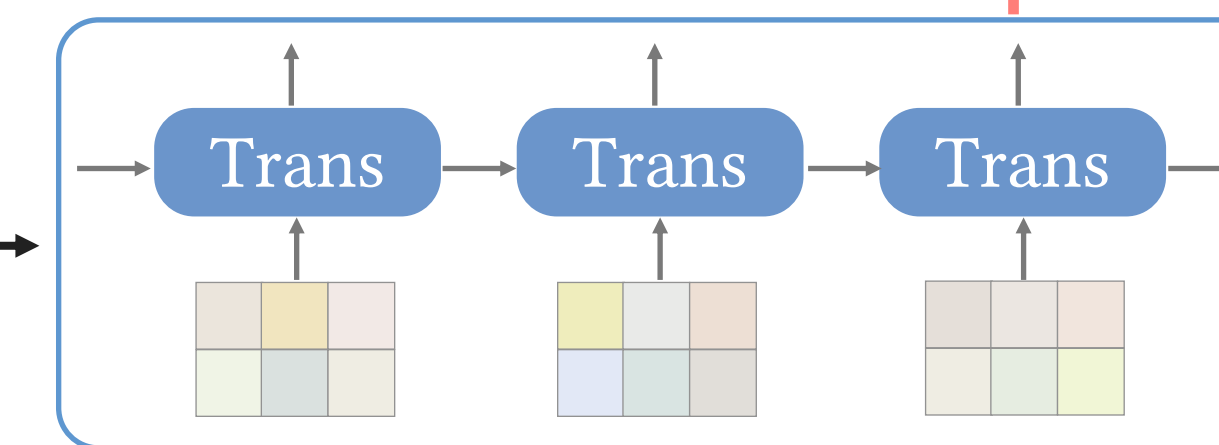


Token Embeddings

Avg Pooling



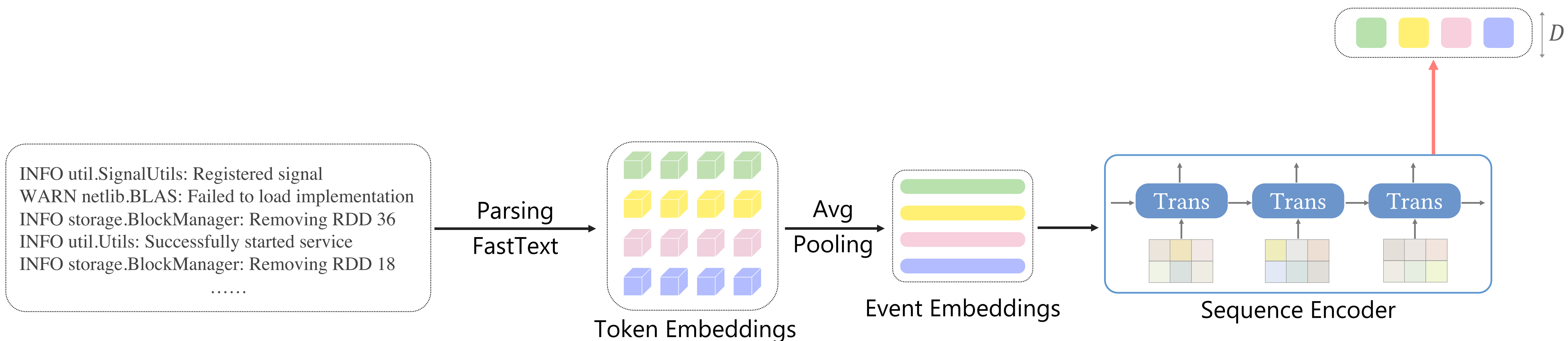
Event Embeddings



Sequence Encoder

# Log Modeling

Log parsing  $\Rightarrow$  Log vectorization  $\Rightarrow$  Log representation learning

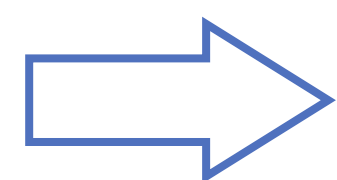




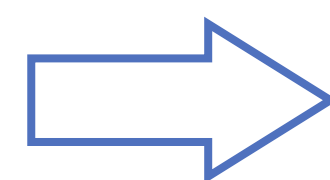


# Aspect-aware Metric Modeling

Pre-processing

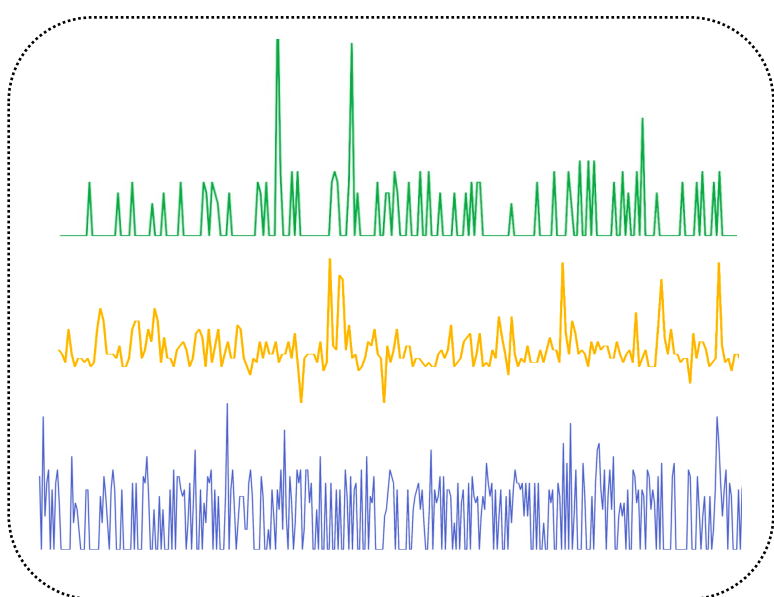


Intra-Aspect Encoder



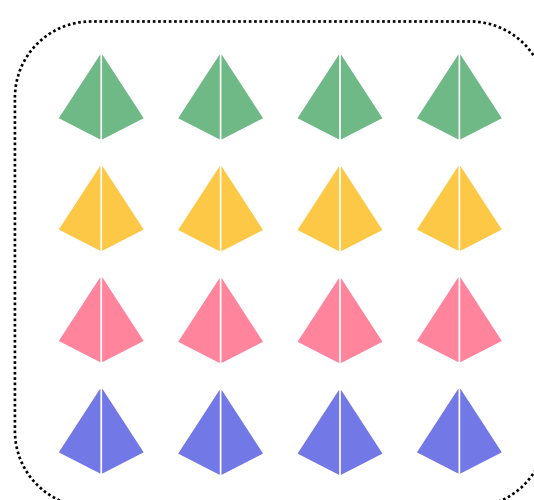
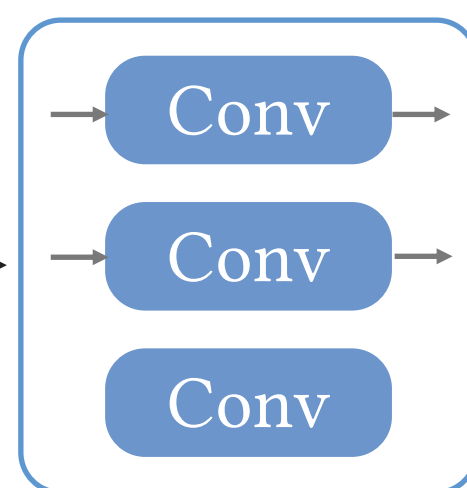
Inter-Aspect Encoder

Multivariate Metrics



Grouping

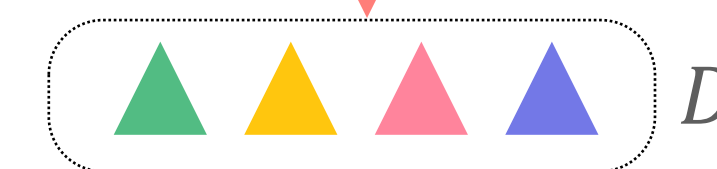
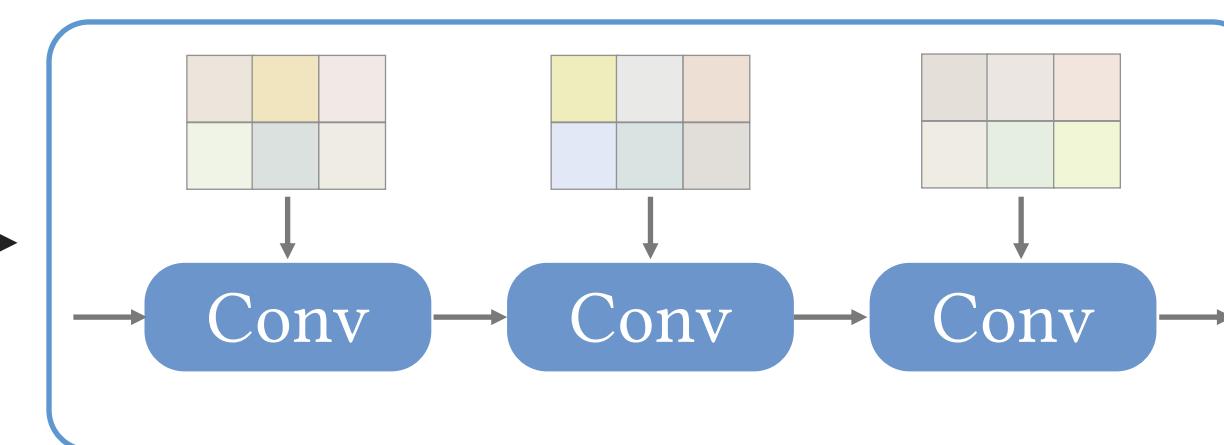
Intra-aspect Encoder



Max Pooling

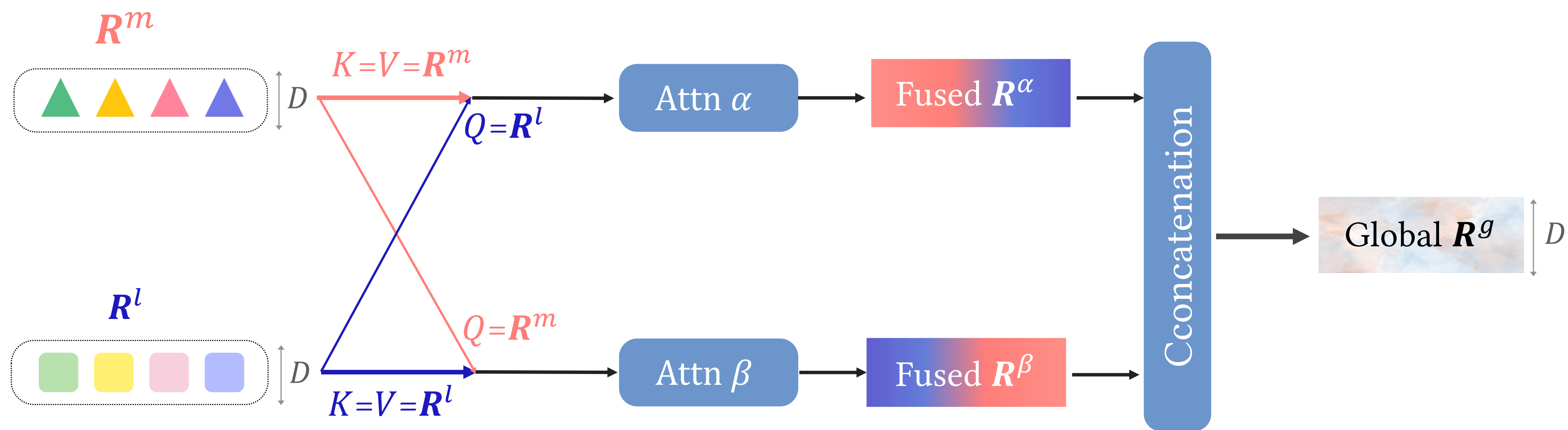


Inter-aspect Encoder



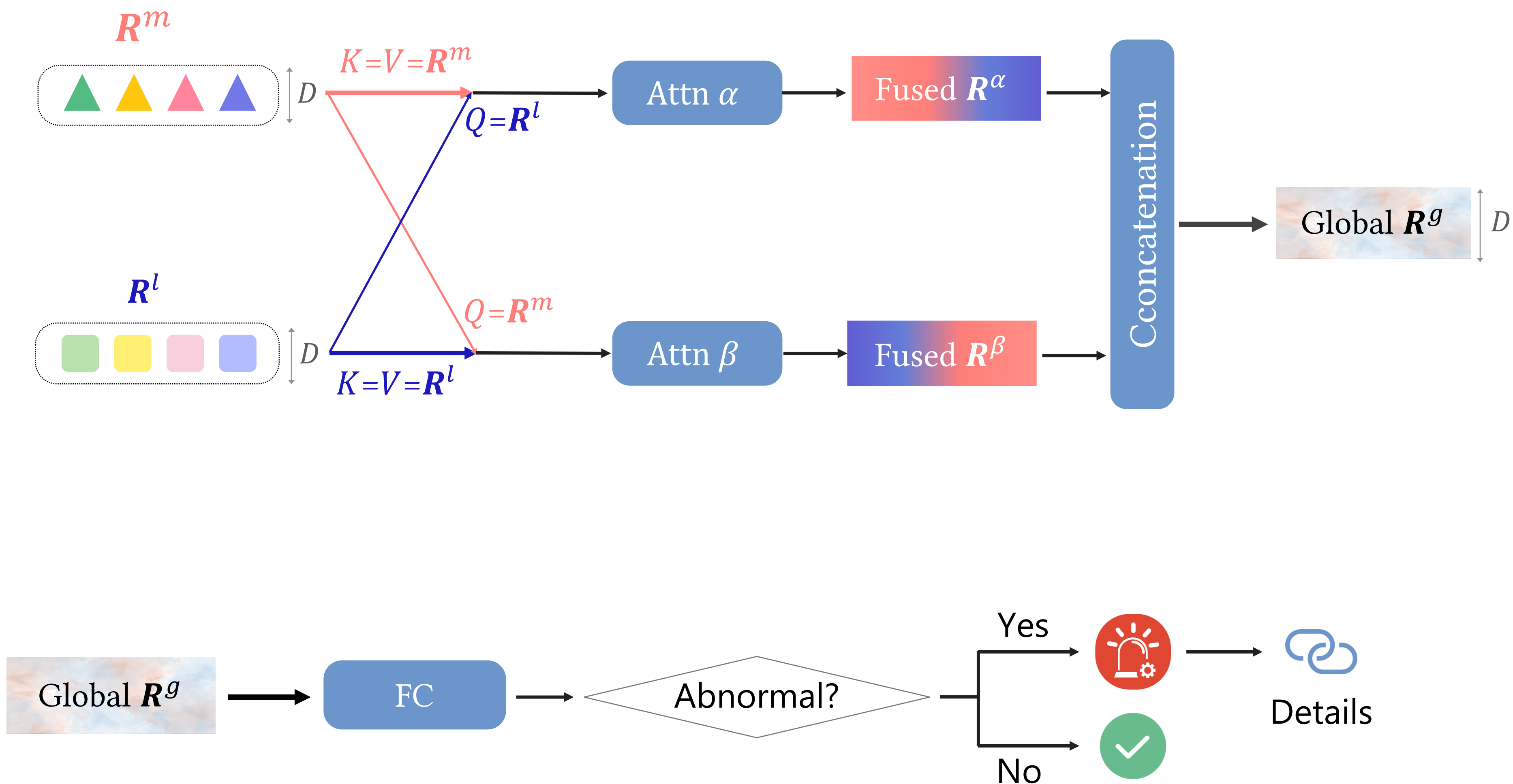


# Heterogeneous Representation Fusion



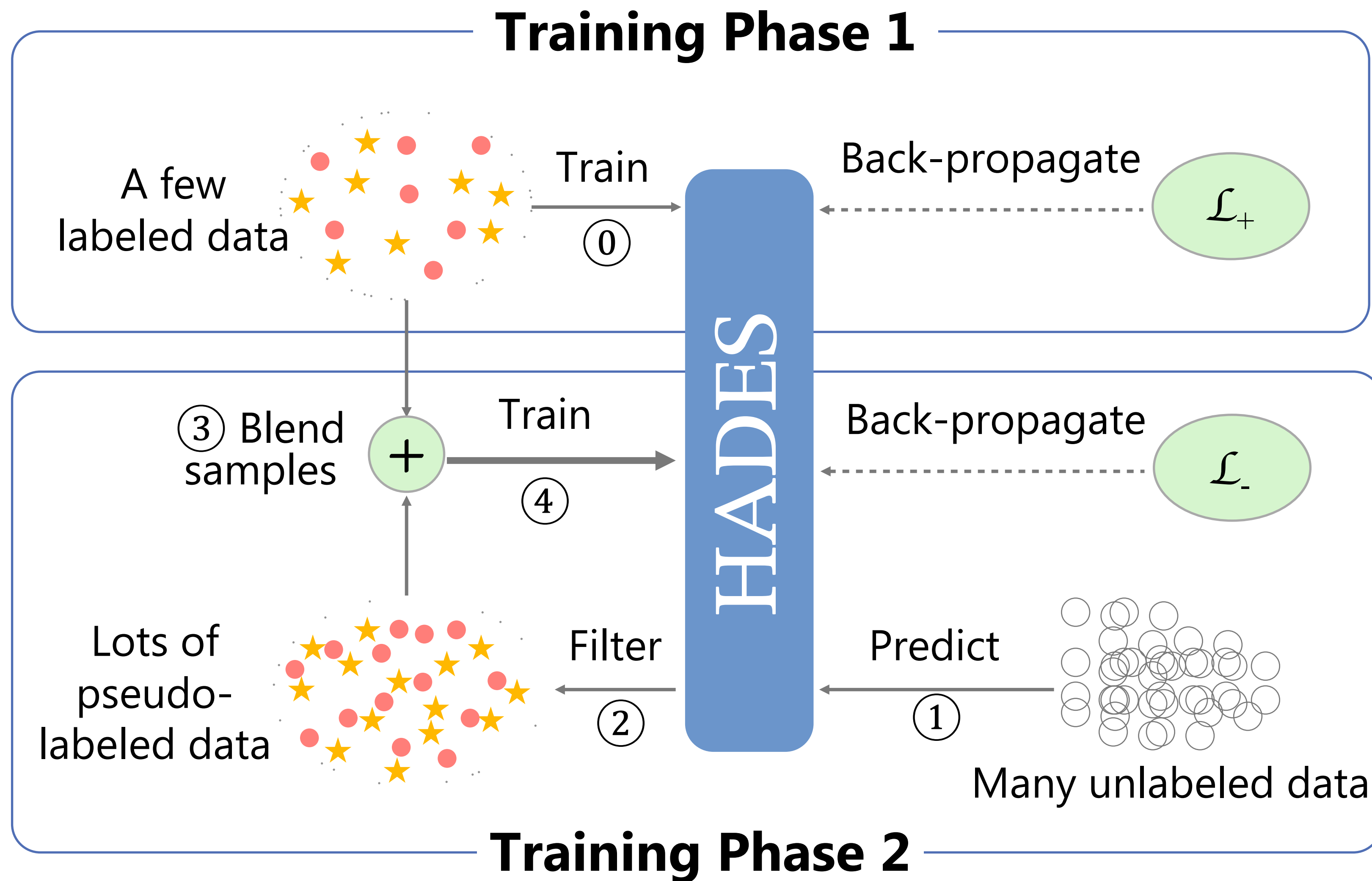


# Detection





# Semi-supervised Learning



### Workload

ID c0d17d481f47bdd9

Status Running

Start at 22/03/01T07:00:00

### Chunk Info

Time 22/03/01T09:28:00 ~ 22/03/01T09:38:00

Status **Abnormal**

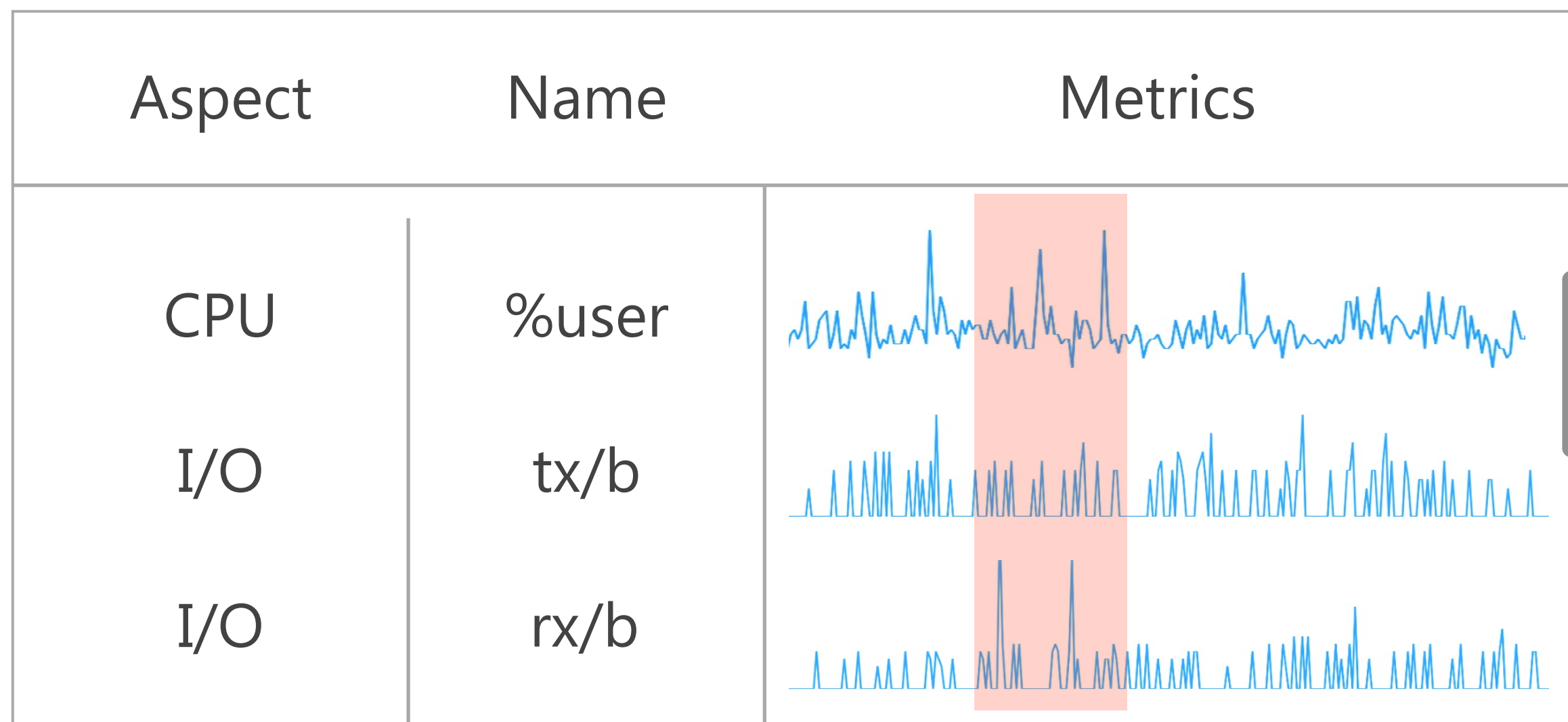
Source Log, Metric

### Log File

Path <http://127.0.0.1/root/workspace/...>

Download

### Key Metrics



### Log Preview

```

INFO storage.BlockManager: Found block rdd_2_3 locally
INFO storage.BlockManager: Found block rdd_2_4 locally
INFO util.SignalUtils: Registered signal
WARN netlib.BLAS: Failed to load implementation
INFO storage.BlockManager: Removing RDD 36
INFO util.Utils: Successfully started service
INFO storage.BlockManager: Removing RDD 18
INFO python.PythonRunner: Times: total = 42, boot = -4131,
init = 4172, finish = 1
    
```



04

## Evaluation

Effectiveness Comparison, Ablation Study...



# How effective is Hades in anomaly detection?

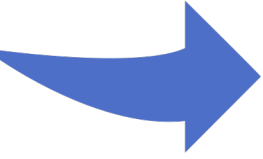
The F1-score of Hades is **9.12%~174.41%** higher than competitors on average.

**Table 1: Overall Performance Comparison.**

Models	Source	Manner	Dataset $\mathcal{A}$			Dataset $\mathcal{B}$			Dataset $\mathcal{C}$		
			<i>F1</i>	<i>Rec</i>	<i>Pre</i>	<i>F1</i>	<i>Rec</i>	<i>Pre</i>	<i>F1</i>	<i>Rec</i>	<i>Pre</i>
SVM- $\mathcal{L}$	Log	Supervised	0.289	0.707	0.181	0.541	0.756	0.421	0.481	0.742	0.356
DeepLog	Log	Unsupervised	0.259	0.386	0.194	0.386	0.526	0.305	0.375	0.524	0.292
PLELog	Log	Semi-supervised	0.314	0.602	0.213	0.463	0.618	0.371	0.434	0.597	0.341
LogRobust	Log	Supervised	0.404	0.684	0.287	0.524	0.718	0.413	0.495	0.698	0.383
SVM- $\mathcal{M}$	Metric	Supervised	0.536	0.833	0.395	0.608	0.839	0.477	0.556	0.801	0.426
Adsketch	Metric	Semi-supervised	0.404	0.476	0.351	0.543	0.644	0.470	0.538	0.649	0.459
OmniAnomaly	Metric	Unsupervised	0.681	0.788	0.601	0.827	0.863	0.794	0.812	0.896	0.743
SRCNN	Metric	Unsupervised	0.342	0.614	0.237	0.467	0.701	0.350	0.472	0.586	0.394
SRCNN-s	Metric	Supervised	0.784	0.826	0.745	0.898	0.938	0.861	0.883	0.926	0.844
SCWarn	Log & Metric	Unsupervised	0.321	0.389	0.273	0.497	0.643	0.405	0.491	0.585	0.423
<b>Hades</b>	Log & Metric	Semi-supervised	<b>0.864</b>	<b>0.870</b>	<b>0.859</b>	<b>0.975</b>	<b>0.984</b>	<b>0.966</b>	<b>0.960</b>	<b>0.969</b>	<b>0.951</b>



# What is the contribution of each design of Hades?



**Table 2: Experimental Results of the Ablation Study.**

<b>Models</b>	Dataset $\mathcal{A}$			Dataset $\mathcal{B}$			Dataset $\mathcal{C}$		
	<i>F1</i>	<i>Rec</i>	<i>Pre</i>	<i>F1</i>	<i>Rec</i>	<i>Pre</i>	<i>F1</i>	<i>Rec</i>	<i>Pre</i>
Hades-supervised	<b>0.866</b>	0.878	0.855	<b>0.979</b>	0.972	<b>0.986</b>	<b>0.961</b>	0.953	<b>0.970</b>
<b>HADES</b>	0.864	0.870	<b>0.859</b>	0.975	<b>0.984</b>	0.966	0.960	0.969	0.951





# How sensitive is Hades to the length of a chunk?

F1.

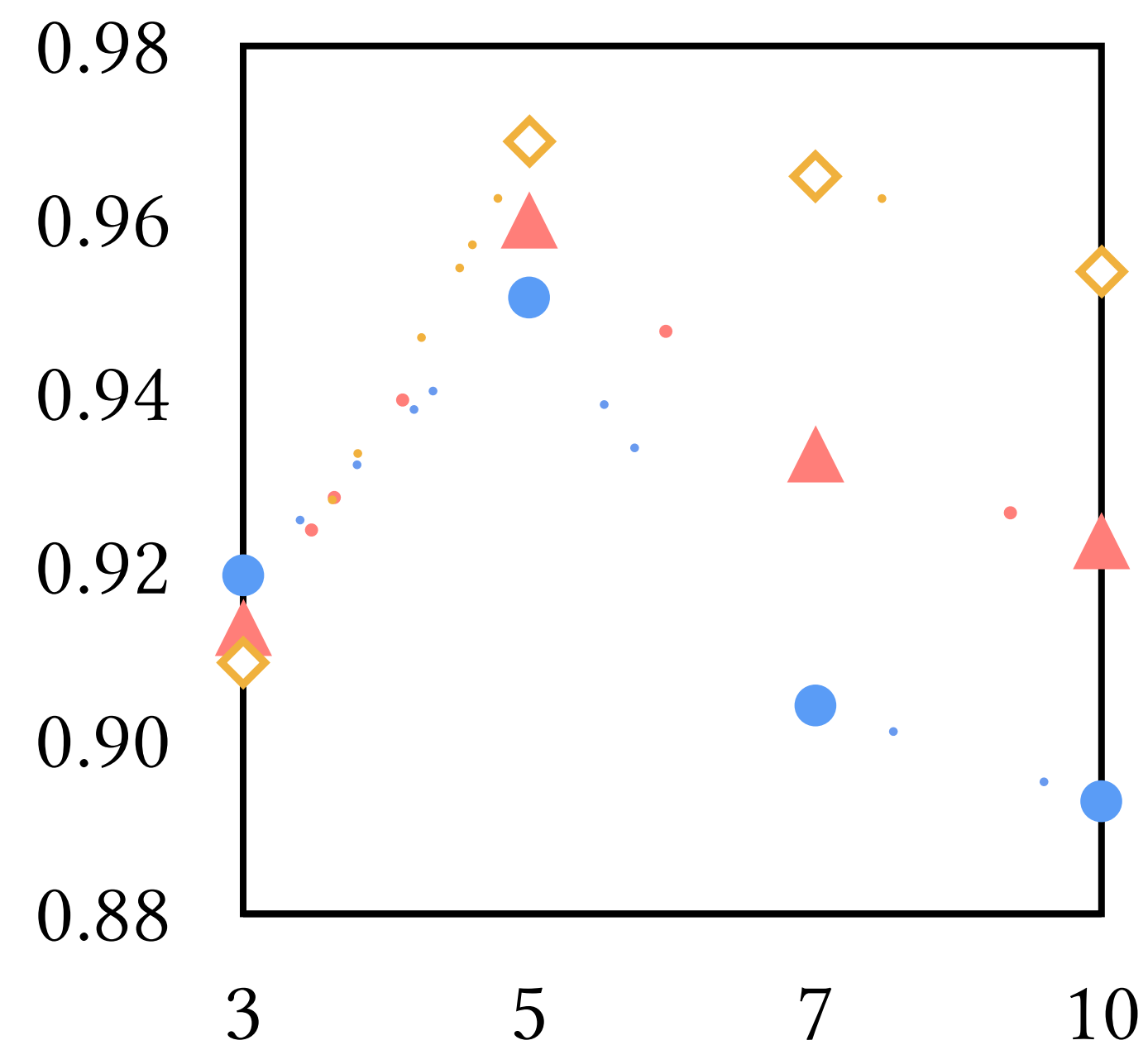
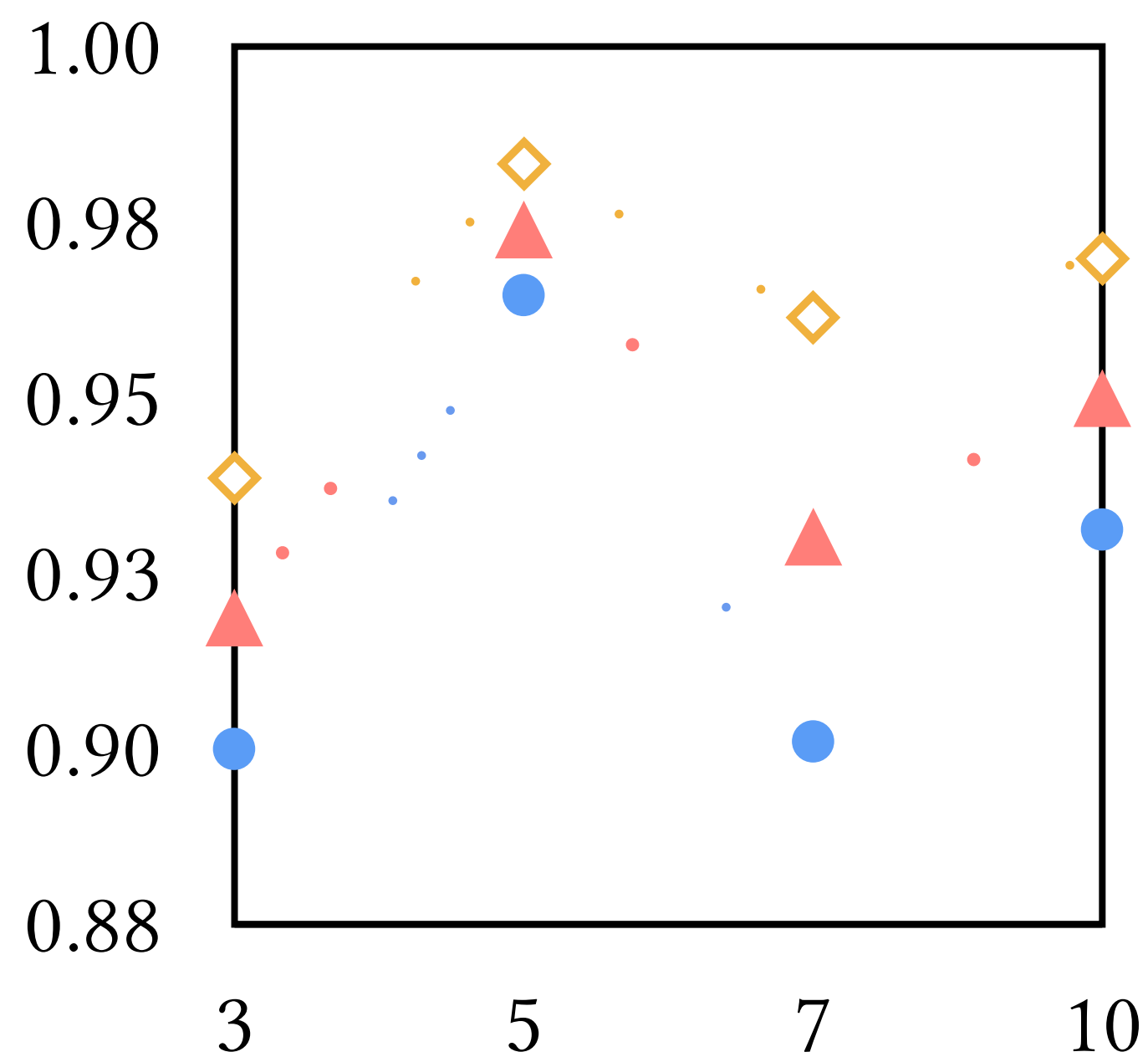
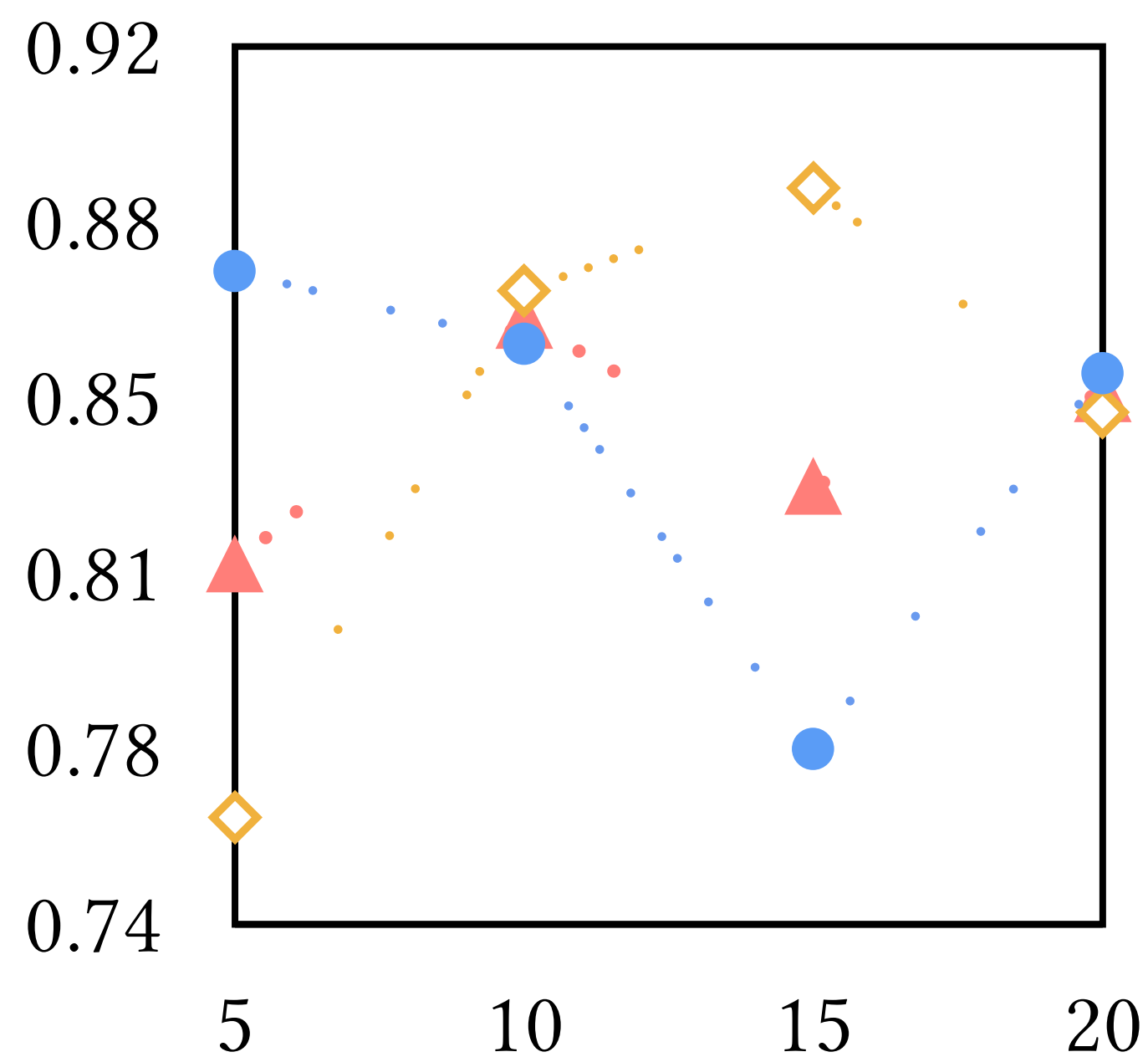
Rec.

Pre.

Dataset  $\mathcal{A}$

Dataset  $\mathcal{B}$

Dataset  $\mathcal{C}$

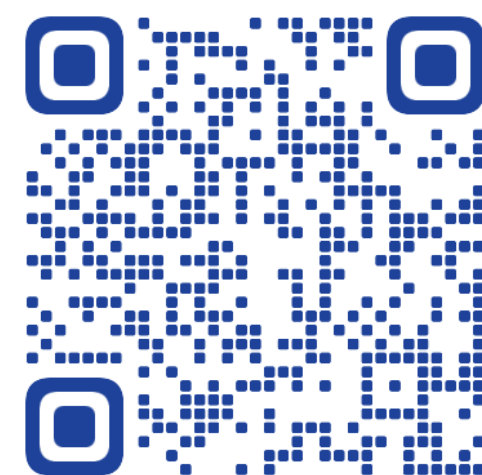


Chunk Length  $T$

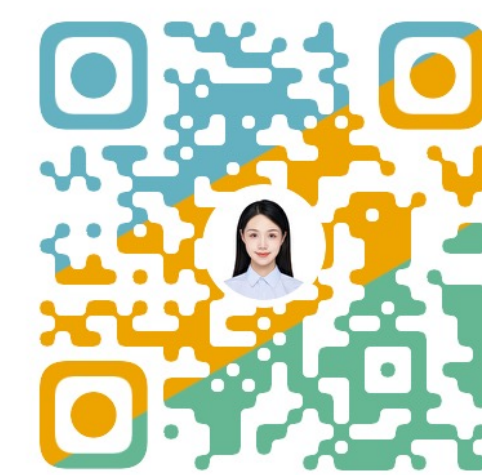


# THANK YOU

Presenter: Cheryl LEE



Full Paper



Home Page