CoorLog: Efficient-Generalizable Log Anomaly Detection via Adaptive Coordinator in Software Evolution

Pei Xiao, Chiming Duan, Minghua He, Tong Jia*, Yifan Wu, Jing Xu, Gege Gao,Lingzhe Zhang, Weijie Hong, Ying Li* and Gang Huang

Peking University
ByteDance

The 40th IEEE/ACM International Conference on Automated Software Engineering (ASE '25, Research Track),

Background

Log anomaly detection (AD) is crucial for ensuring system reliability.

The Problem with Software Evolution:

Log Evolution: Frequent software updates change log structures and patterns.

- **Log Entry Evolution**: changes in individual log *(Case 1)*
- **Log Sequences Evolution**: changes in order or dependencies *(Case 2)*

Case 1 Log Entry Evolution

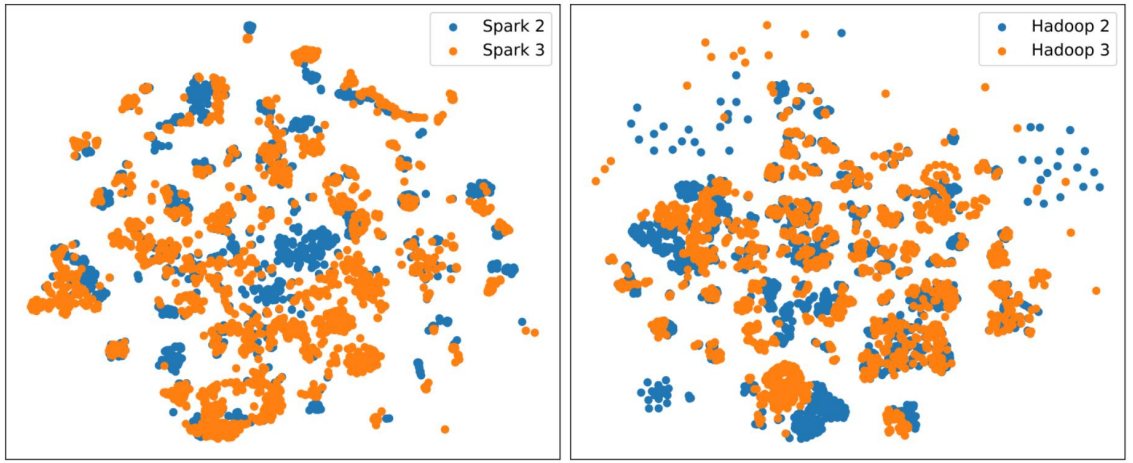
Spark 2	Started reading broadcast variable <*>
Spark 3	Started reading broadcast variable <*> with <*> pieces(estimated total size <*> MiB)

Case 2 Log Sequences Evolution

Spark 2	E1 → E3
Spark 3	E1 → E2 → E3

E1: Connecting to driver: <*>
E2: Successfully registered with driver
E3: Resources for <*>:

Fig.1 The case of log entry and sequences evolution



(a) Spark (b) Hadoop

Fig.2 The t-SNE visualization of log ebbemding distribution changes from Spark 2 to 3 and from Hadoop 2 to 3

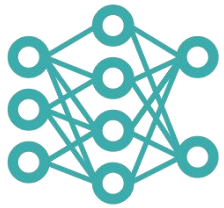
Challenges

How to trade-off Efficiency and Generalization ?

Small Model

Compact Parameters

- Cost-effective
- Bad generalization

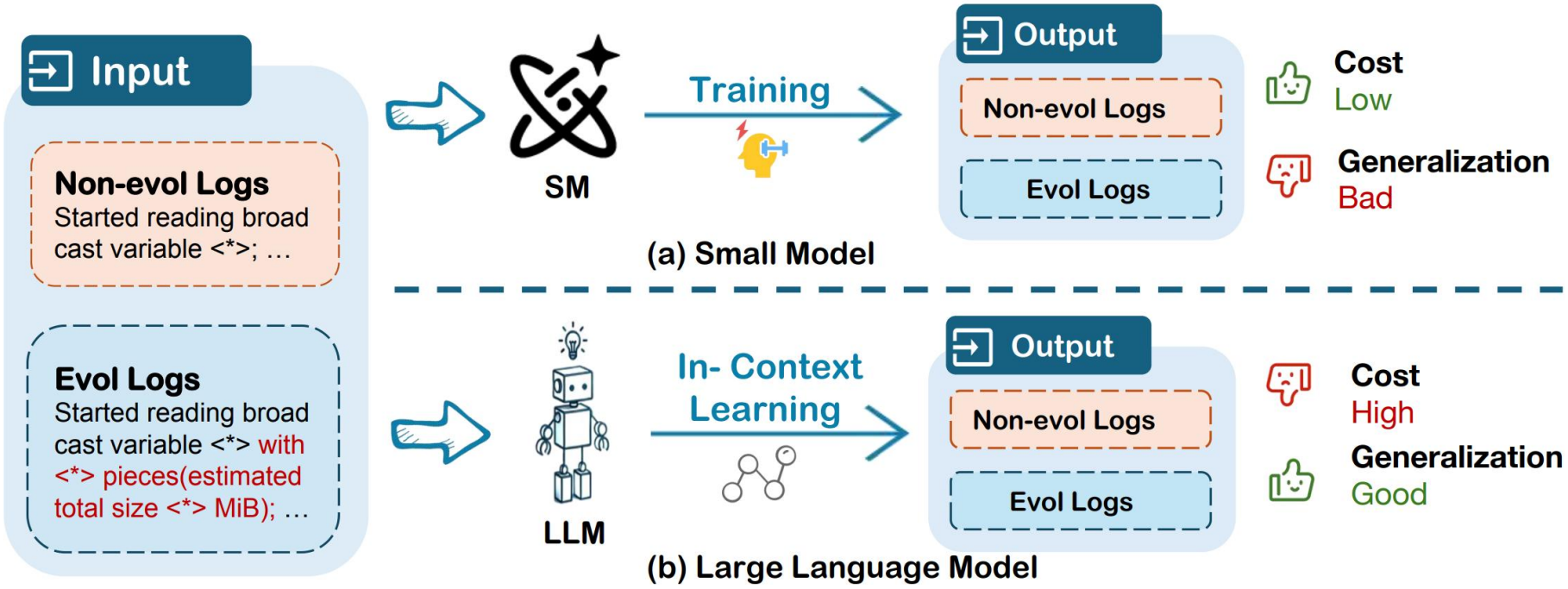


VS

Large Language Model

Massive Parameters

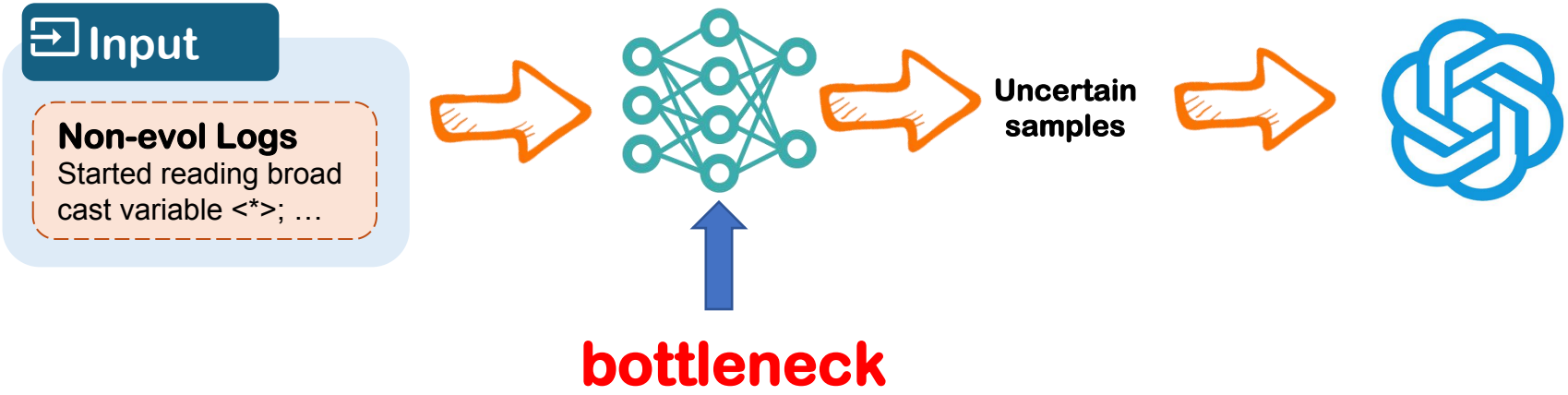
- High Cost
- Good Generalization



Note: "Evol logs" refers to evolved logs.

Challenges

In existing cascade architectures, the reliance still lies on the preceding small model.



[1] Adaptivelog: An adaptive log analysis framework with the collaboration of large and small language model.

[2] CLSLog: Collaborating Large and Small Models for Log-based Anomaly Detection.

[3] LogRAG: Semi-Supervised Log-based Anomaly Detection with Retrieval-Augmented Generation.

Proposed Solution

Trade-off Efficiency and Generalization Via Adaptive Coordinator

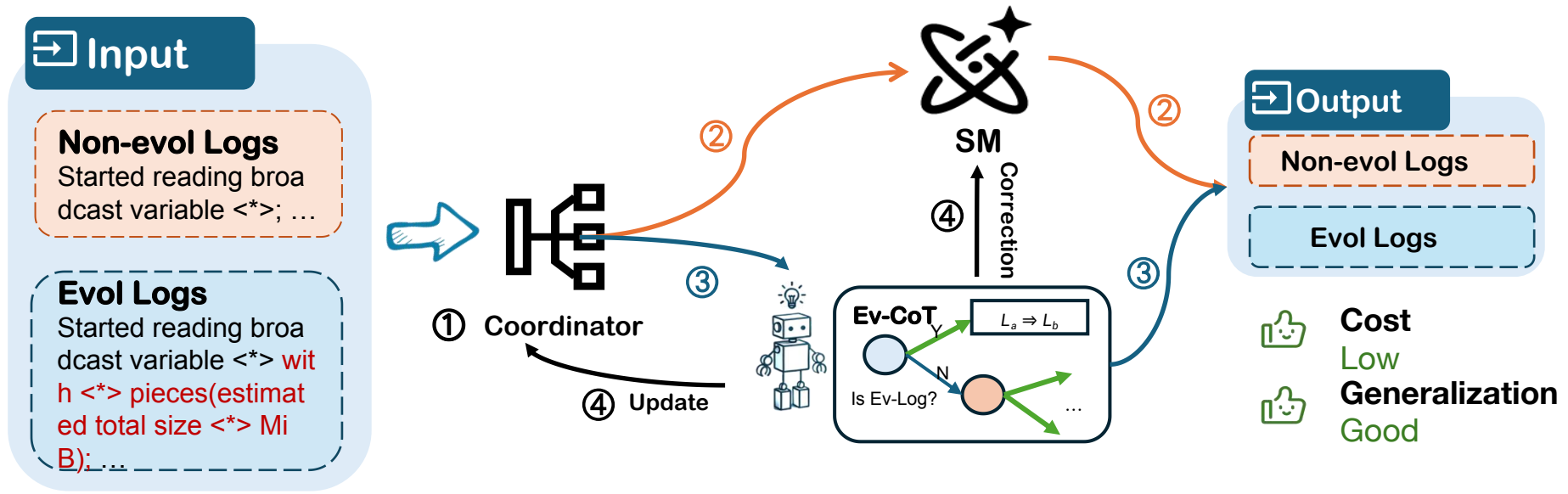
Collaborating LLM and SM

SM

High efficiency for in-distribution samples (non-evolved logs)

LLM

Deep semantic analysis for out-of-distribution samples (evolved logs)



We introduce an Adaptive Coordinator that routes logs to the appropriate model and continuously adapt to software evolution

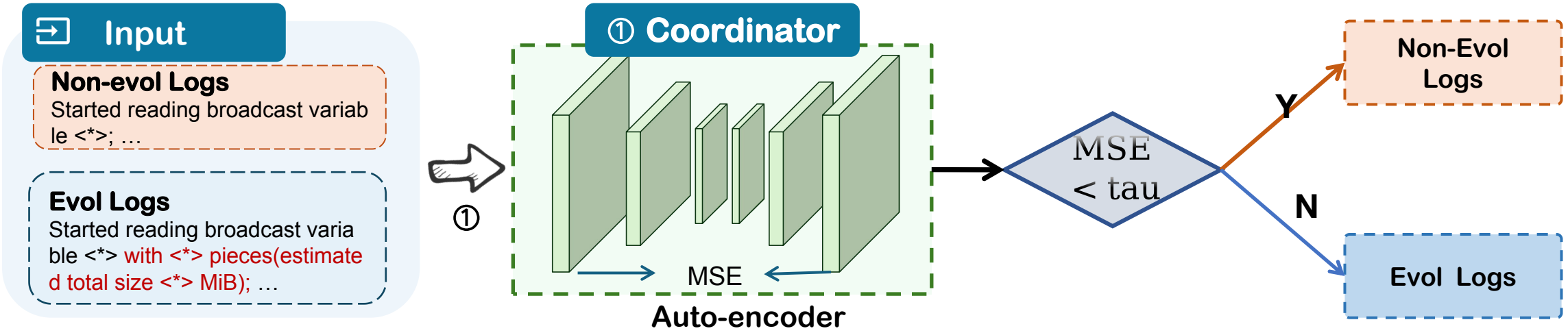
Stage 1: Identifying Logs by Coordinator

Use an AutoEncoder to distinguish between known and new log patterns

1. Training: trianed by pre-evolution logs to learn original distribution

2. Inference: Identifying if incoming logs out-of- distribution by reconstruction error

- *If the MSE is high, the log is flagged as "Evolved."*
- *If the MSE is low, the log is flagged as "No-evolved."*



Stage 2: Efficient AD for Non-Evol Logs

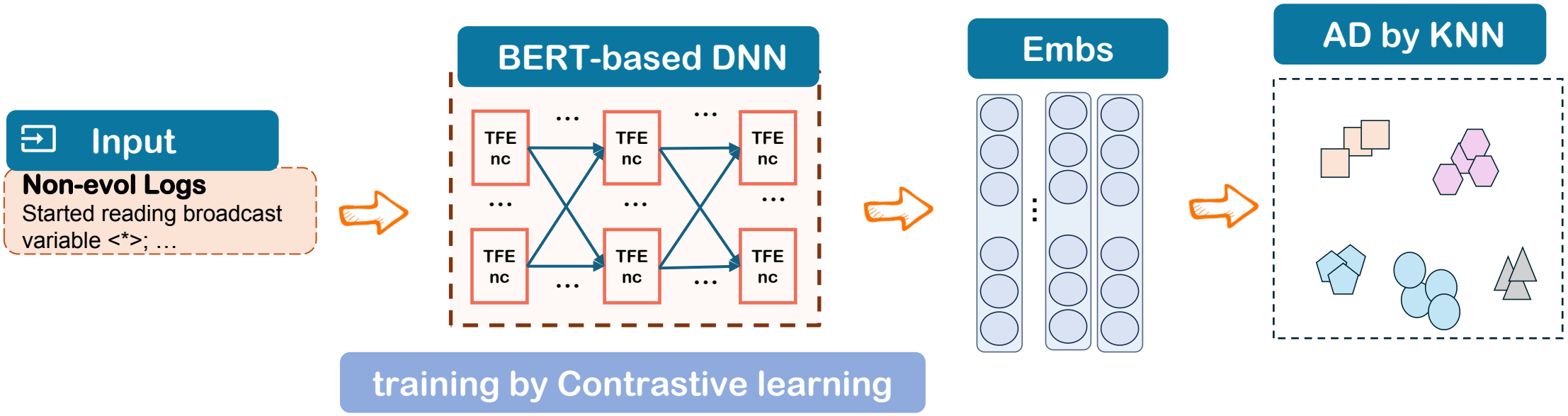
Use a BERT-based Network to process in-distribution logs

model as a semantic similarity task to avoid issuse(false alarms) caused by log parsing

1. Training: Fine-tuned with contrastive learning

2. Detecting: Anomaly detection based on embeddings extracted from SM through similarity

- *Retrieve the Top-k most similar samples*
- *Perform detection using similarity-weighted voting*



Stage 3: Detect Evolved Logs by LLM

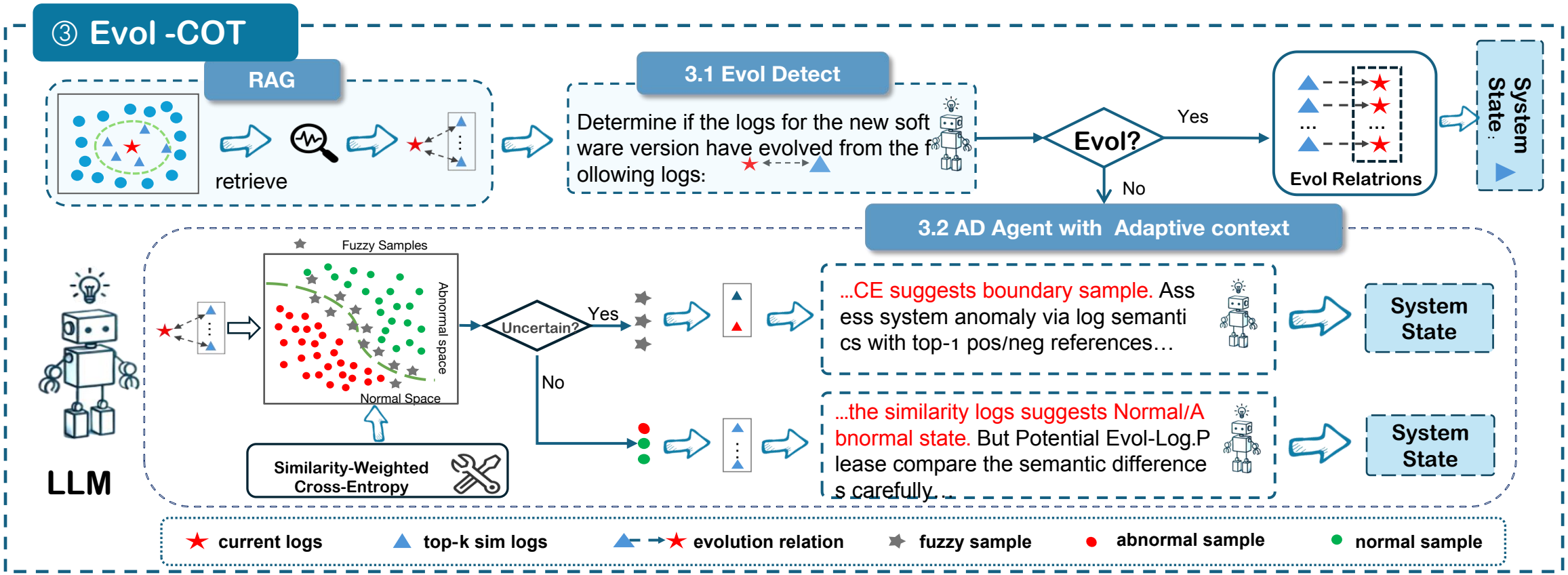
Evol-CoT framework for fine-grained inference on evolved logs

(1) Evol Detect :

Retrieves the Top-K most similar logs.
Detecting if an evolutionary relationship exists

(2) AD Agent : Based on Log Semantics

Using uncertainty to dynamically adjust the detection strategy



Stage 4: Adaptive Evolution Mechanism

Use AEM to avoid redundant inference for the same samples in LLM

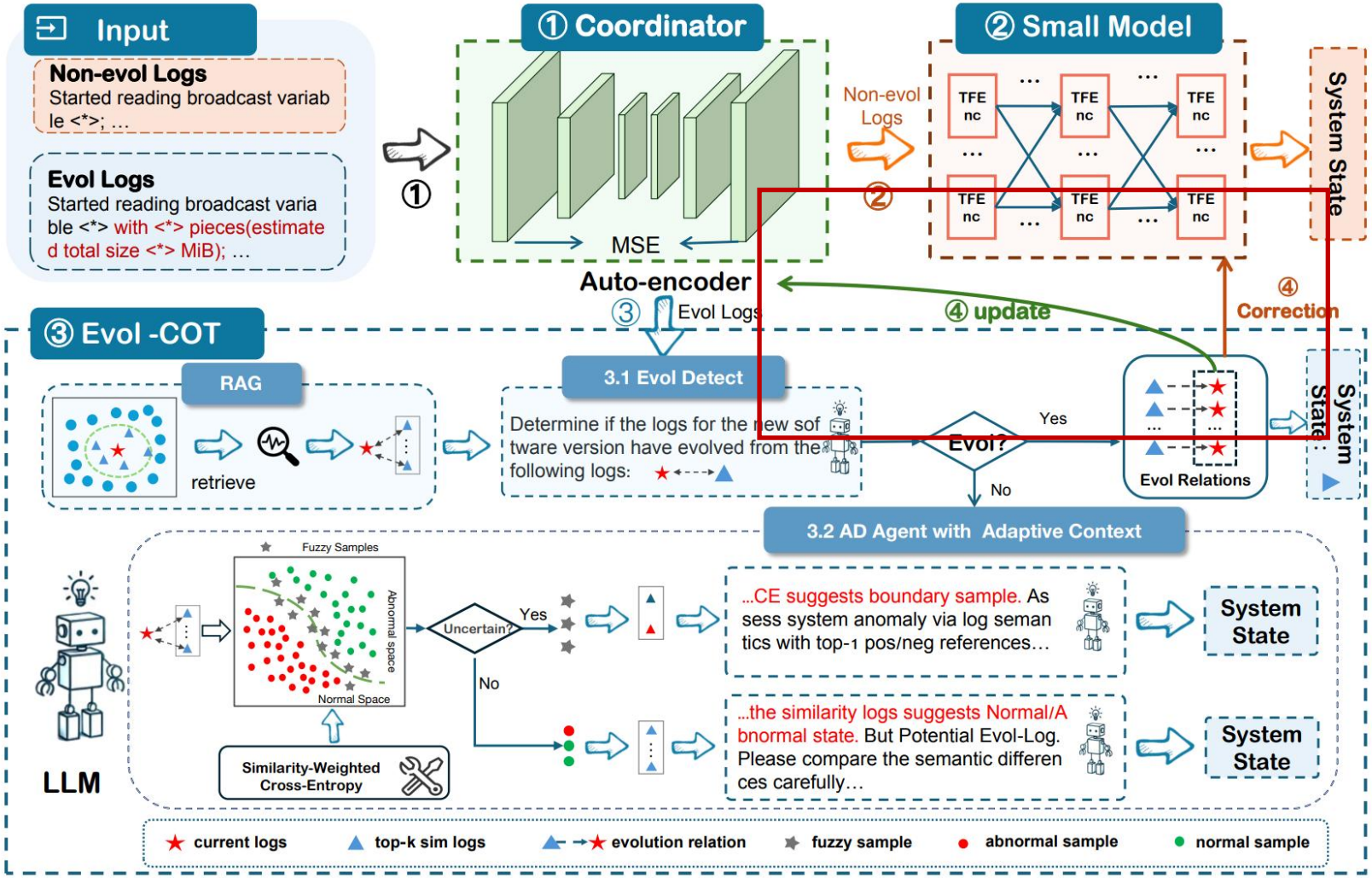
(1) Update Coordinator

evolved logs of identified Evol relationship are collected to fine-tune the autoencoder using the reconstruction loss

(2) Correcting SM

use identified Evol relationship (si, sj) as a similar sample set to fine-tune SM using the contrastive loss

Since evolutionary relationship identification is a simple semantic classification task, LLM performs well, making its results highly reliable.



Experiment and Result

Key Result (1) Accuracy & Generalization

CoorLog significantly outperforms all baselines (DL-based, LLM-based) , demonstrating robust generalization (Table I).

LogEvol-Hadoop												
	Intra-version						Inter-version					
	Hadoop 2 → Hadoop 2			Hadoop 3 → Hadoop 3			Hadoop 2 → Hadoop 3			Hadoop 3 → Hadoop 2		
Method	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1
LogSed	0.910	**0.995**	0.951	0.925	0.986	0.955	0.371	0.988	0.540	0.390	0.993	0.560
DeepLog	0.913	0.985	0.947	0.926	**1.000**	0.961	0.386	**0.999**	0.556	0.410	0.971	0.576
LogAnomaly	0.926	0.994	0.958	0.939	0.988	0.963	0.389	0.998	0.560	0.407	**0.995**	0.578
BERT	0.928	0.731	0.817	0.959	0.837	0.894	0.865	0.706	0.778	0.952	0.763	0.847
LogRobust	0.935	0.981	0.957	0.948	0.983	0.965	0.782	0.824	0.803	0.813	0.846	0.829
LogBERT	0.941	0.977	0.959	0.953	0.987	0.970	0.875	0.852	0.863	0.898	0.871	0.884
LogOnline	0.948	0.984	0.966	0.963	0.989	0.976	0.893	0.895	0.894	0.913	0.908	0.911
LLMeLog	0.952	0.967	0.959	0.963	0.975	0.969	0.912	0.923	0.917	0.928	0.934	0.931
EvLog	0.945	0.982	0.963	0.952	0.988	0.970	0.770	0.941	0.847	0.857	0.913	0.884
Our	**0.993**	0.968	**0.980**	**0.997**	0.982	**0.990**	**0.946**	0.983	**0.964**	**0.994**	0.957	**0.975**
LogEvol-Spark												
	Intra-version						Inter-version					
	Spark 2 → Spark 2			Spark 3 → Spark 3			Spark 2 → Spark 3			Spark 3 → Spark 2		
Method	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1
LogSed	0.842	0.914	0.877	0.907	0.923	0.915	0.013	0.917	0.026	0.010	0.914	0.020
DeepLog	0.862	**0.952**	0.905	0.858	**0.976**	0.914	0.017	**0.947**	0.032	0.014	0.909	0.026
LogAnomaly	0.931	0.939	0.935	0.898	0.947	0.922	0.020	0.923	0.038	0.017	**0.948**	0.034
BERT	0.943	0.750	0.835	**1.000**	0.684	0.812	0.550	0.696	0.615	**1.000**	0.568	0.715
LogRobust	0.949	0.837	0.889	0.974	0.857	0.912	0.732	0.753	0.742	0.934	0.783	0.851
LogBERT	0.948	0.875	0.909	0.973	0.886	0.927	0.805	0.813	0.809	0.949	0.822	0.881
LogOnline	0.954	0.904	0.928	0.986	0.899	0.940	0.843	0.864	0.853	0.957	0.875	0.914
LLMeLog	0.959	0.920	0.938	0.950	0.857	0.900	0.840	0.799	0.828	0.762	0.934	0.851
EvLog	0.970	**0.974**	**0.972**	0.944	0.888	0.915	0.922	0.700	0.795	0.920	0.812	0.863
Our	**0.976**	0.932	0.954	0.979	0.918	**0.947**	**0.904**	0.855	**0.879**	0.968	0.904	**0.935**

Table I Comparison of different strategies

Key Result (2) Efficiency & Cost

CoorLog reduces processing time by 91.63% and token consumption by 85.59% compared to vanilla. (Table II)

Metric	Method	Spark 2 → 3	Hadoop 2 → 3
Calls	w/o coord	4,246	34,302
	Our	289	2,174
Token (k)	w/o coord	9,048.85	21,347.73
	Our	1,557.30	2,480.65
Time (ms)	LogRobust	0.96	0.43
	LogAnomaly	1.69	0.83
	Our_{SM}	0.23	0.04
	w/o coord	810.10	724.61
	Our	78.40	54.30

Table II Comparison of efficiency and computational cost

Conclusion: CoorLog achieves high accuracy with low cost, making it suitable for real-world deployment.

Ablation Study

Key Result (3) Effect of Coordinator

The small model performs poorly on Evol logs but performs well on Non-evol logs (Table III), and the curve of the key parameter τ aligns with expectations (Fig. 3-a).

Logs	Method	Spark 2 → 3			Hadoop 2 → 3		
		Pr	Re	F1	Pr	Re	F1
Non-evol	Our_{SM}	0.950	0.901	0.925	0.985	0.979	0.982
	+AEM	0.935	0.925	0.930	0.955	0.943	0.949
Evol	Our_{SM}	0.468	0.417	0.441	0.375	0.402	0.387
	+AEM	0.640	0.610	0.625	0.537	0.489	0.512
ALL	Vanilla	0.785	0.209	0.330	0.705	0.518	0.595
	Evol-CoT	0.829	0.913	0.870	0.944	0.885	0.914
	Our_{SM}	0.698	0.848	0.766	0.893	0.861	0.876
	+AEM	0.721	0.867	0.794	0.895	0.872	0.883
	Our	0.904	0.855	0.879	0.946	0.983	0.964

Table III Ablation study on different log categories

Key Result (4) Effect of Small Model

The key hyperparameter k is selected based on the small model's performance on the validation set (Fig. 3-b).

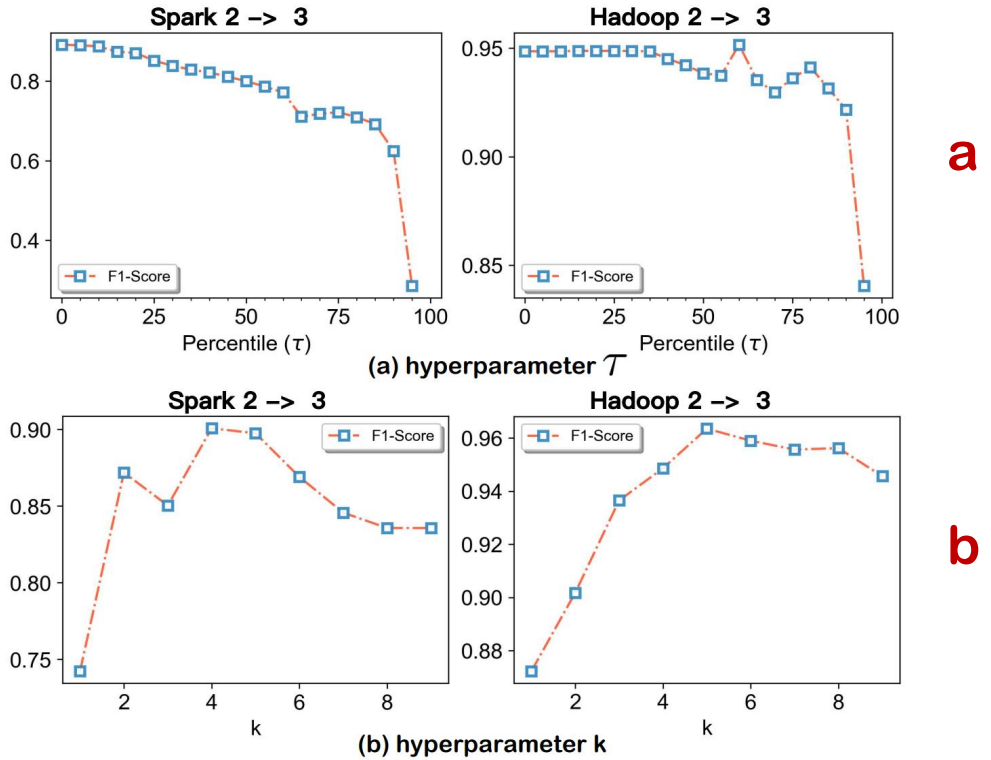


Fig 3 ablation study on parameters

Conclusion:The coordinator can identify concept drift logs.

Ablation Study

Key Result (5) Effect of Evol-CoT

Evol-CoT'outperforms Vanilla and small models (Table III-a)

Evol-CoT providing **interpretability** (Fig. 4)

Key Result (6) Effect of AEM

Evol Detect only requires LLM for semantic parsing, achieving high accuracy (Table IV)

AEM further reduces unnecessary LLM usage without compromising accuracy (Table III-b)

	Logs	Method	Spark 2 → 3			Hadoop 2 → 3		
			Pr	Re	F1	Pr	Re	F1
b	Non-evol	Our_{SM}	0.950	0.901	0.925	0.985	0.979	0.982
		+AEM	0.935	0.925	0.930	0.955	0.943	0.949
a	Evol	Our_{SM}	0.468	0.417	0.441	0.375	0.402	0.387
		+AEM	0.640	0.610	0.625	0.537	0.489	0.512
		Vanilla	0.785	0.209	0.330	0.705	0.518	0.595
		Evol-CoT	0.829	0.913	0.870	0.944	0.885	0.914
	ALL	Our_{SM}	0.698	0.848	0.766	0.893	0.861	0.876
		+AEM	0.721	0.867	0.794	0.895	0.872	0.883
		Our	0.904	0.855	0.879	0.946	0.983	0.964

Table III Ablation study on different log categories

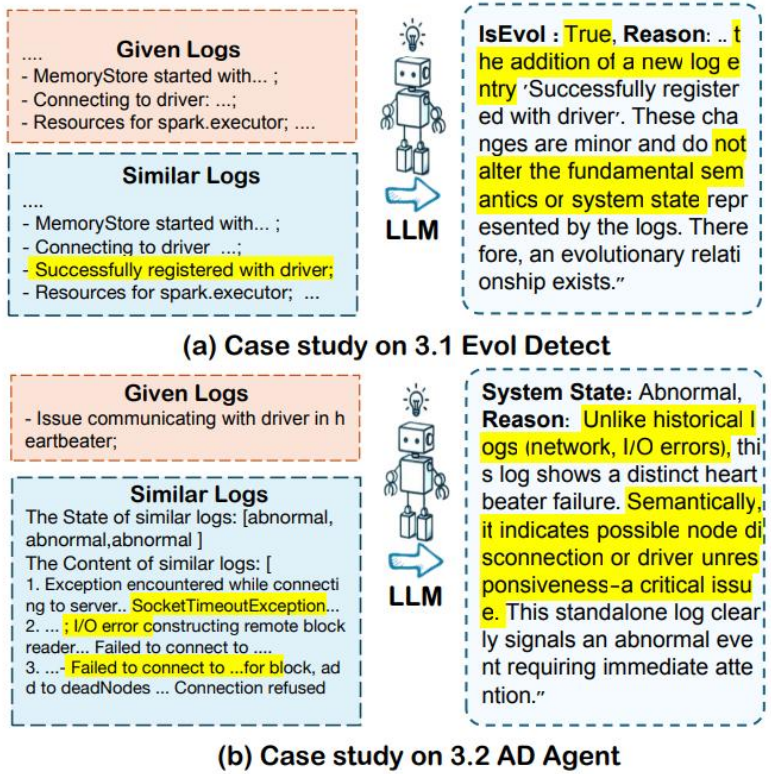


Fig 4 Case studies

	#Relations	Accuracy	AEM Time (ms)
Spark 2 → 3	288	0.993	0.18
Hadoop 2 → 3	541	0.986	0.02

Table IV Statistics of evolution relations

Conclusion & Future Work

Conclusion

- **Problem:** Small models fail to generalize to evolved logs, while large models are too inefficient.
- **Solution:** CoorLog uses an adaptive coordinator to efficiently combine the efficiency of SMs with the generalization power of LLMs.
- **Impact:** Our framework achieves a superior balance of high accuracy and low computational cost, providing a robust solution for log anomaly detection in evolving software.

Future Work

- Explore different combinations of Small and Large Language Models.
- Validate the CoorLog framework across a wider range of industrial systems and scenarios.

Thank you for listening!

Q&A

Pei Xiao

xiaopei@stu.pku.edu.cn

Peking University