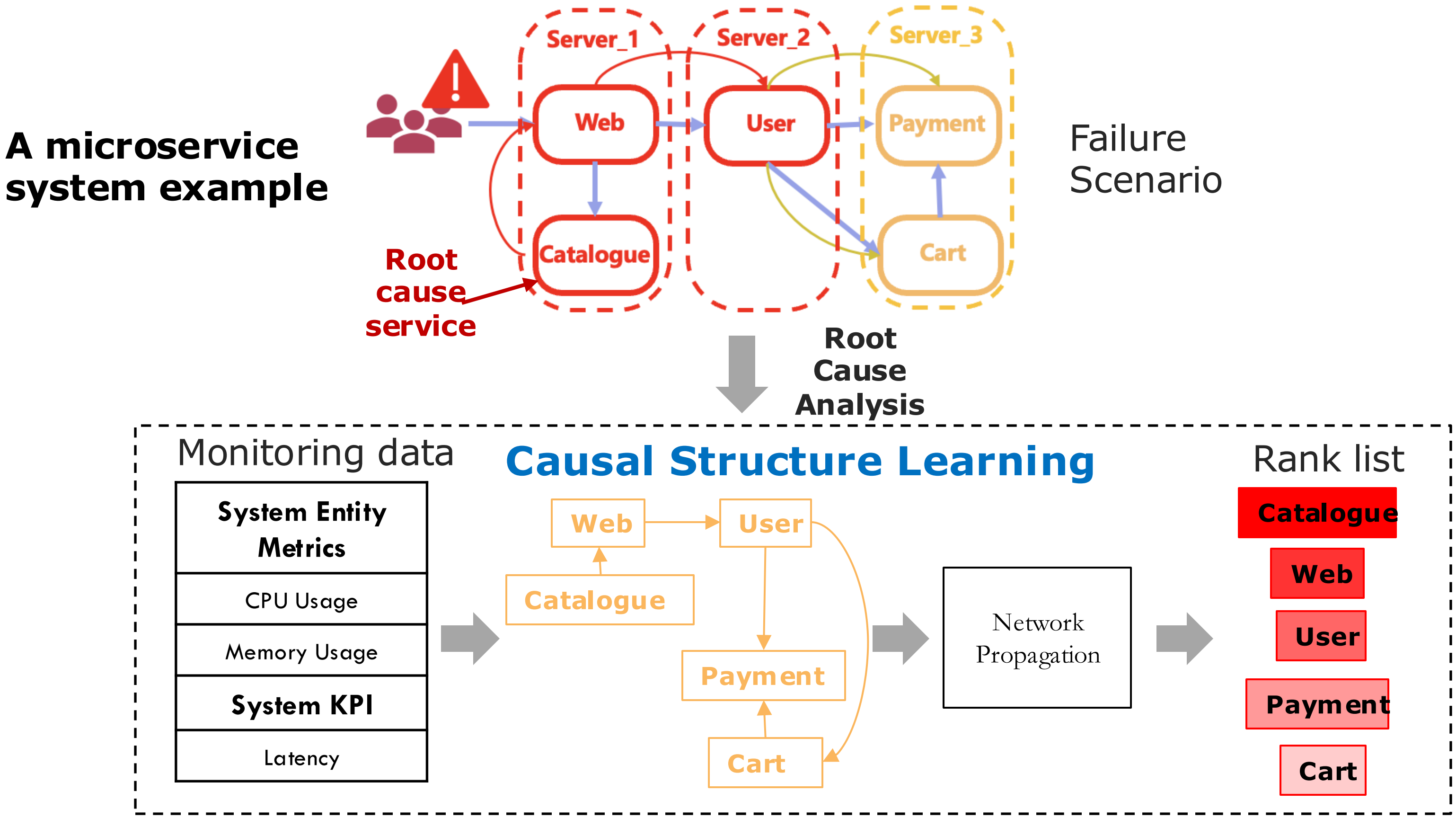
Incremental Causal Graph Learning for Online Root Cause Analysis

Authors: Dongjie Wang, Zhengzhang Chen, Yanjie Fu,
Yanchi Liu, Haifeng Chen

NEC Laboratories America

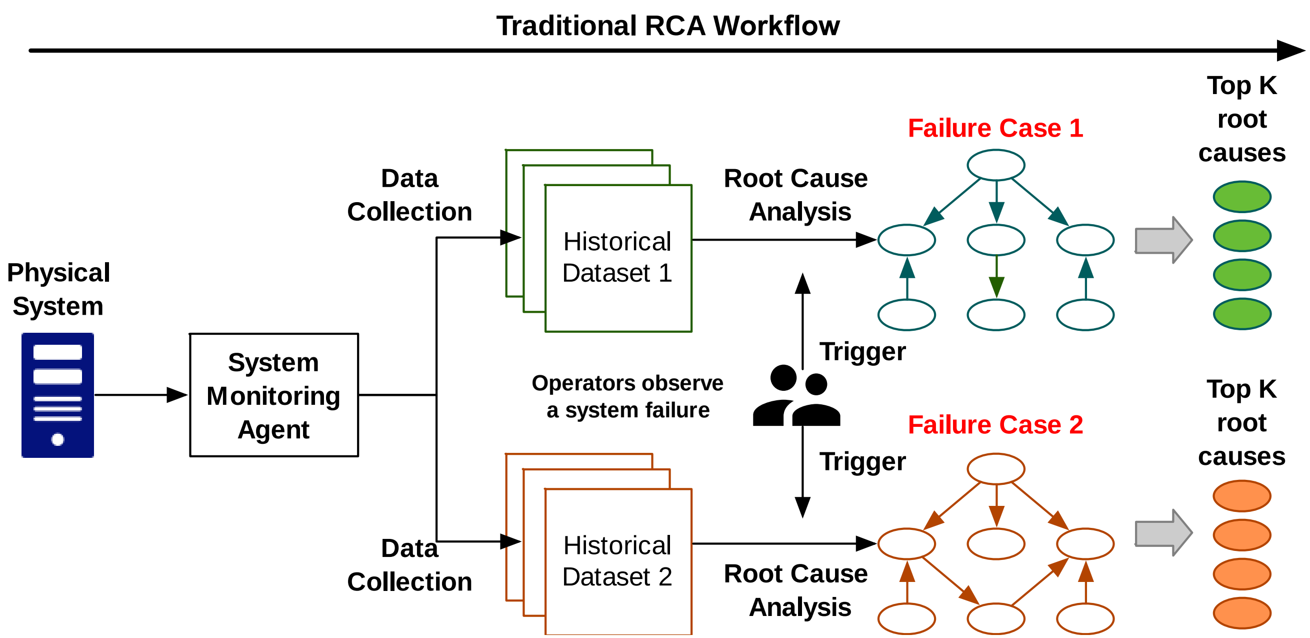
Background: Root Cause Analysis (RCA)



- Input: System entity metrics and system KPI (i.e., multi-variate time series)
- Output: Top-k possible root causes (i.e., malfunctional system entities)

Automatic locating root causes based on large-scale system monitoring data

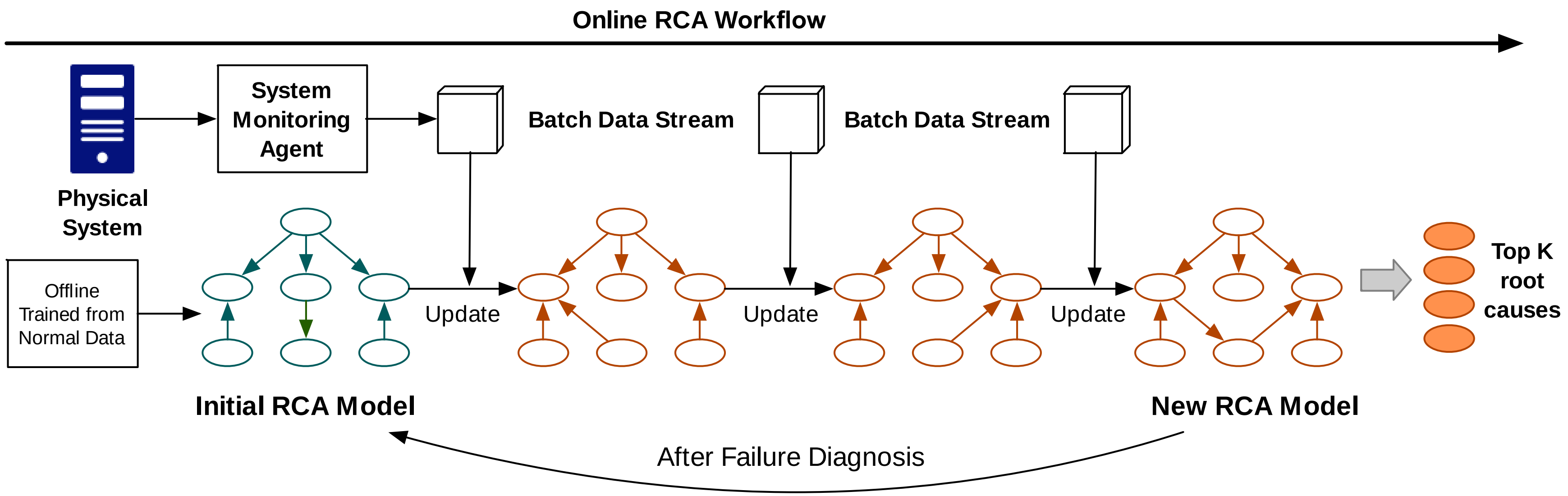
Limitations of Traditional Offline RCA



- ## Limitations
  - **Inefficient:** For a new system failure, need to retrain/rebuild the model from scratch
  - **Slow:** For a large-scale system, often require a long data collecting time and RCA running time
  - **Strict assumption:** The collected data should be only related to one failure case or one system state; Hard to determine which time period of data should be used for training the model

Traditional RCA is slow, inefficient and constrained

Workflow of Online RCA



- Online RCA can **incrementally learn** the change of the RCA model
- Online setting **"virtually" accelerates** RCA process by leveraging the learned **invariant dependencies**
- Online learning can **mitigate damages/losses** by triggering early RCA

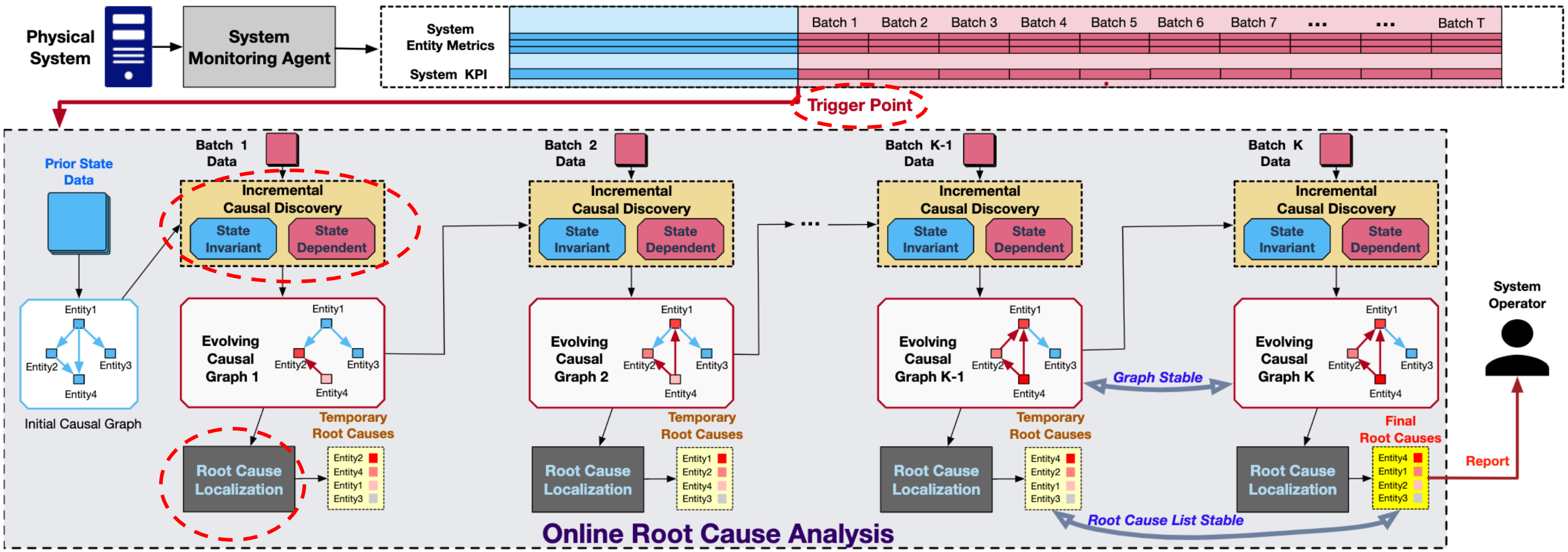
Online RCA is more efficient and practical compared with traditional offline RCA

Two Straightforward Online RCA Methods

- **Option 1: Directly use the offline trained RCA model**
  - Suffered by different data distributions. **(Low RCA performance)**
  - Lack of knowledge about new system failures. **(Missing domain-specific failure patterns)**
- **Option 2: Keep updating the RCA model for each incoming data batch**
  - Too many useless model updates. **(High computational costs)**
  - Include too much noisy data (i.e., data may belong to multiple system phrases/failure cases). **(Poison RCA performance)**
- **An efficient incremental RCA framework should be proposed for accurately locating root causes in time**

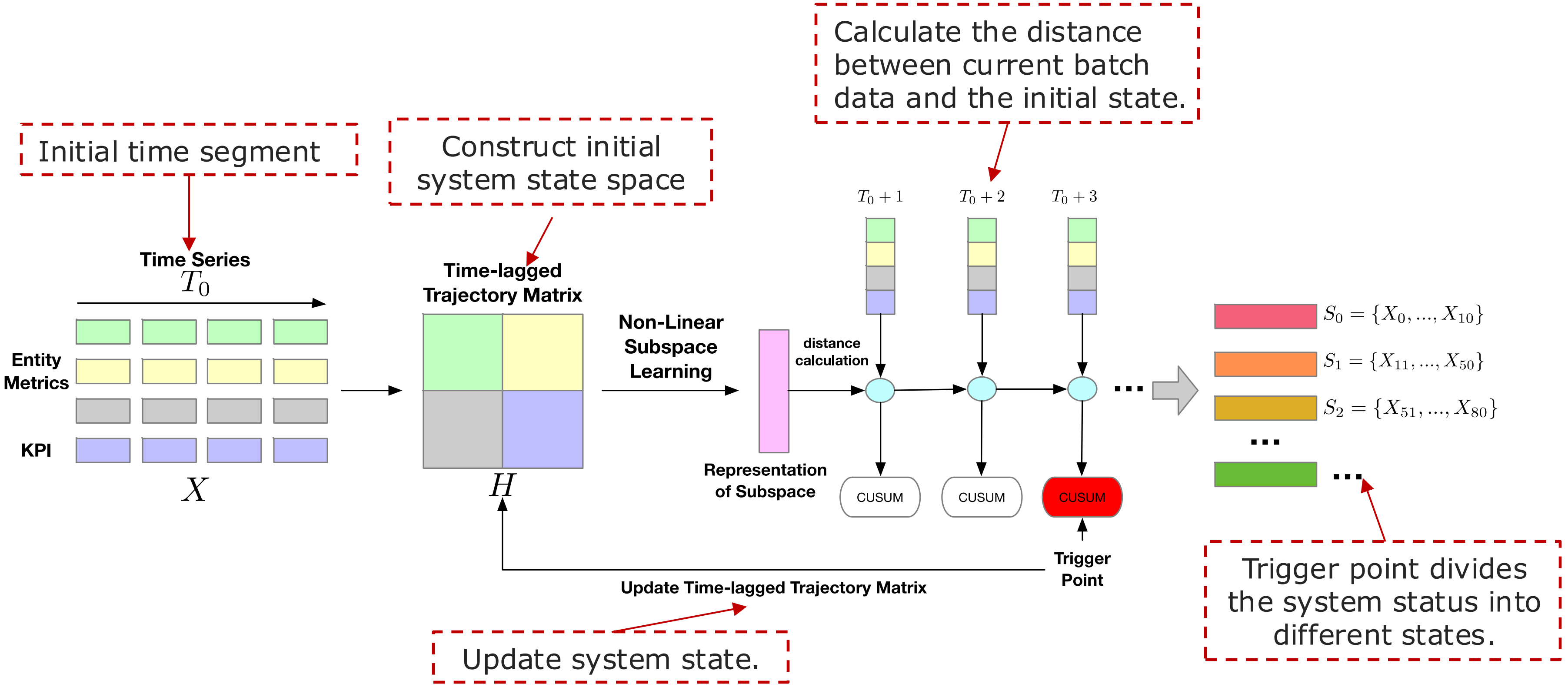
When and how to update model is critical for online RCA

CORAL Framework Overview



- **Online Trigger Point Detection**
  - Automatically detect system state change and trigger incremental causal discovery
- **Incremental Causal Discovery**
  - Integrate system state invariant and system state dependent information for incrementally constructing causal graph
- **Root Cause Localization**
  - Localize root causes of system failures using the learned causal structure

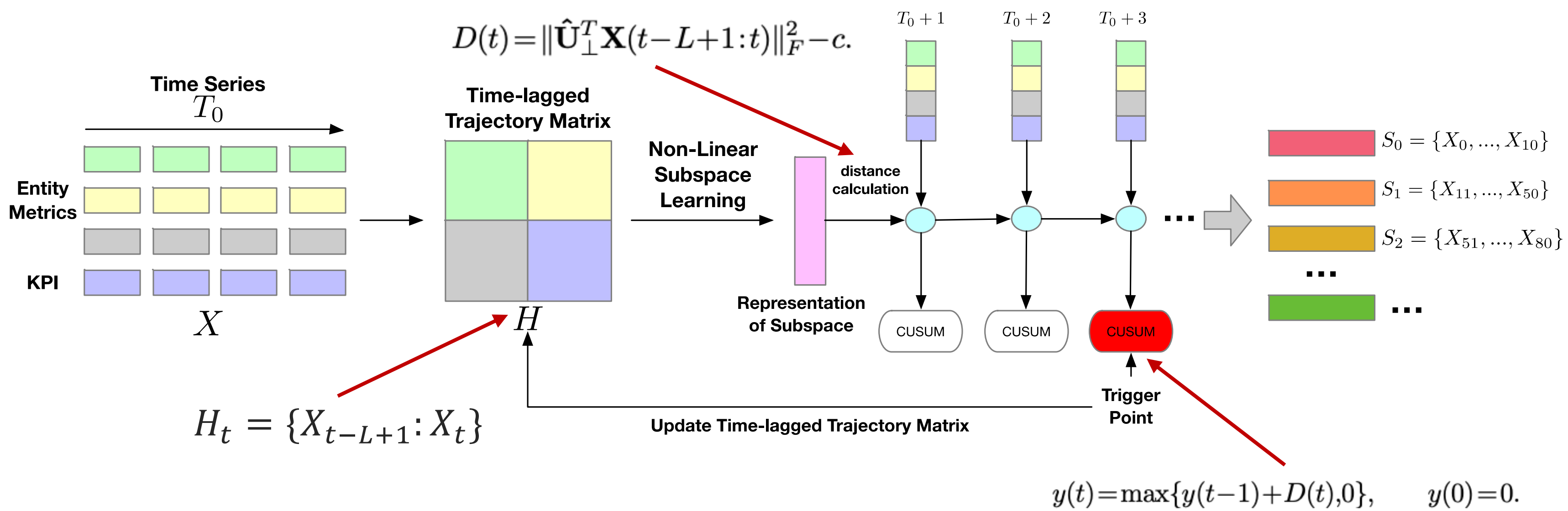
Online Trigger Point Detection



- Detect system state changes in a short delay time.
- Computational cost should be low.

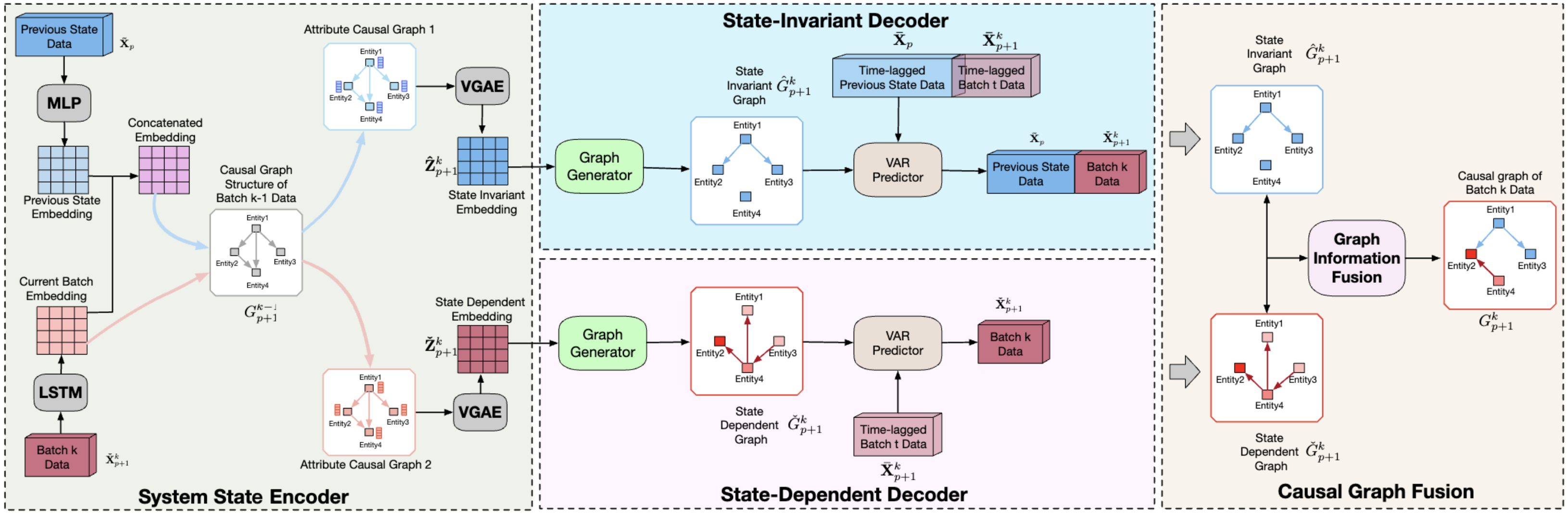
Online trigger point detector should detect state change in a short delay time

Online Trigger Point Detection



- **Distance Function:** The distance will remain small as long as the observations continue to follow the same latent time series
- **CUSUM Score Calculation:** The algorithm uses the subspace distance as a detection score to construct a CUSUM statistic to perform the sequential hypothesis test.

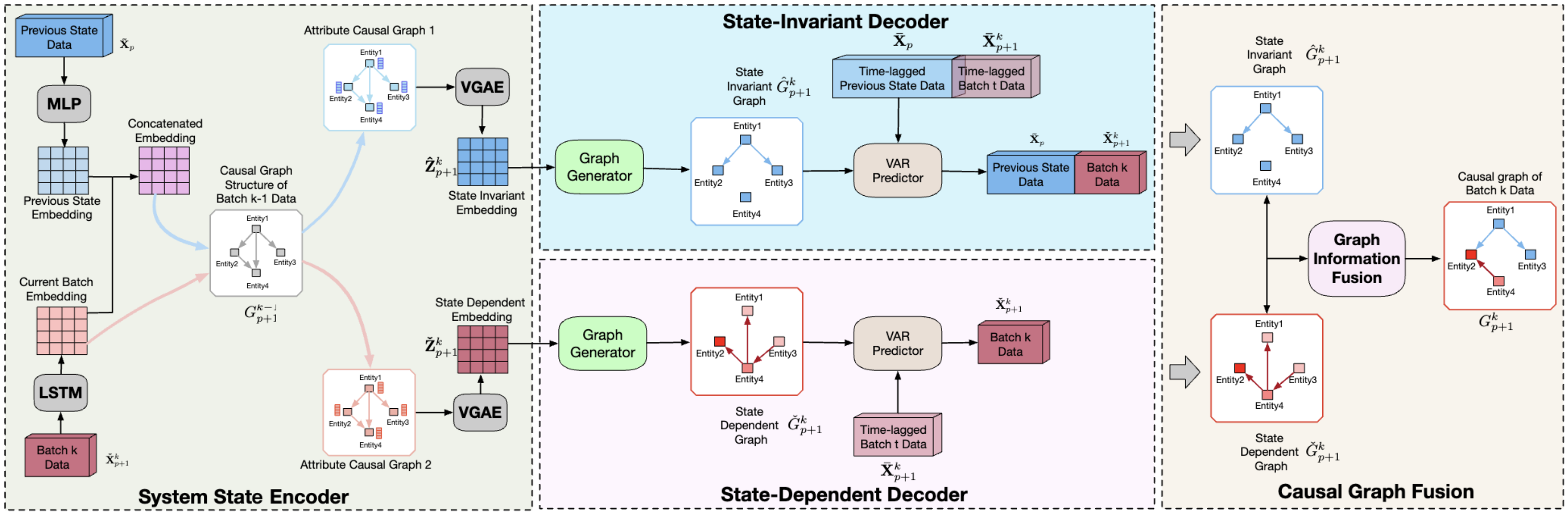
Disentangled Causal Graph Learning



- **Intuition**
  - **Dynamic Causal Relationships**
    - Relations between variables and their temporal dynamics may depend on the system state
  - **Inherent Stable Causal Relationships**
    - Some inherent system dependencies will never vary over time

Causal relationships between system entities can be complex and vary over time

Disentangled Causal Graph Learning

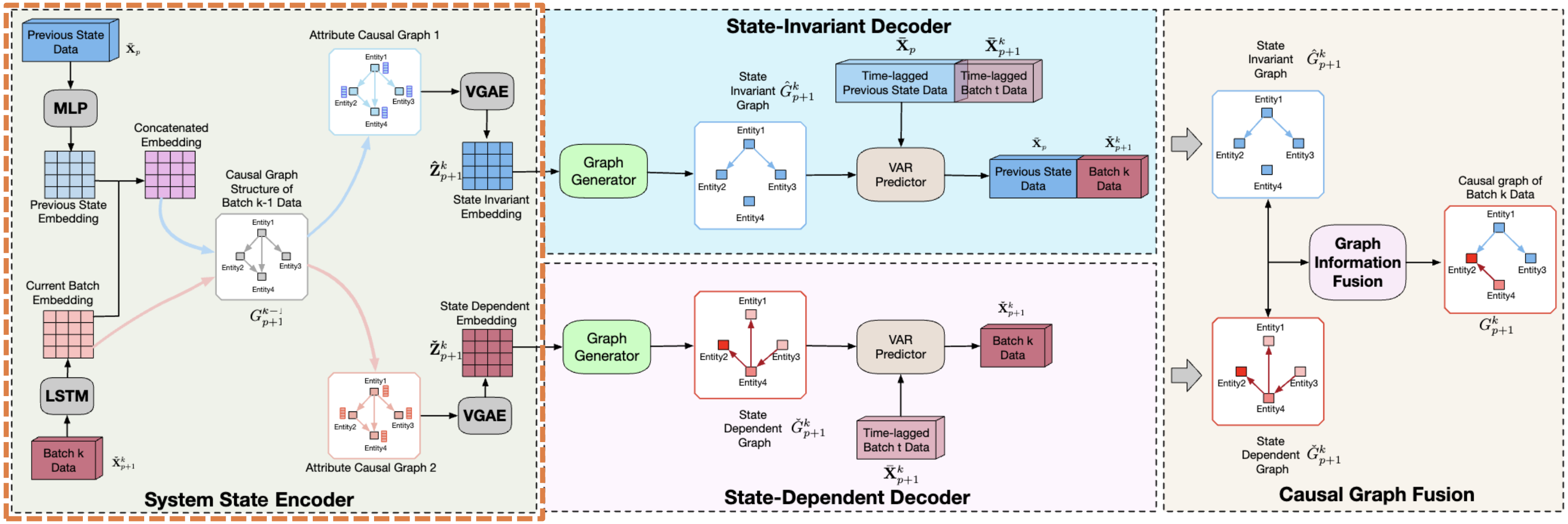


Goals

- **Learn state-invariant causal graph**
  - Keeps inherent unvarying causal relations for disentangling state-invariant information
- **Learn state-dependent causal graph**
  - Captures time-varying causal relations for disentangling state-dependent information

Disentangle state-invariant and state-dependent causal relationships

Disentangled Causal Graph Learning



System State Encoder

- Integrate the previous system state and current data batch information

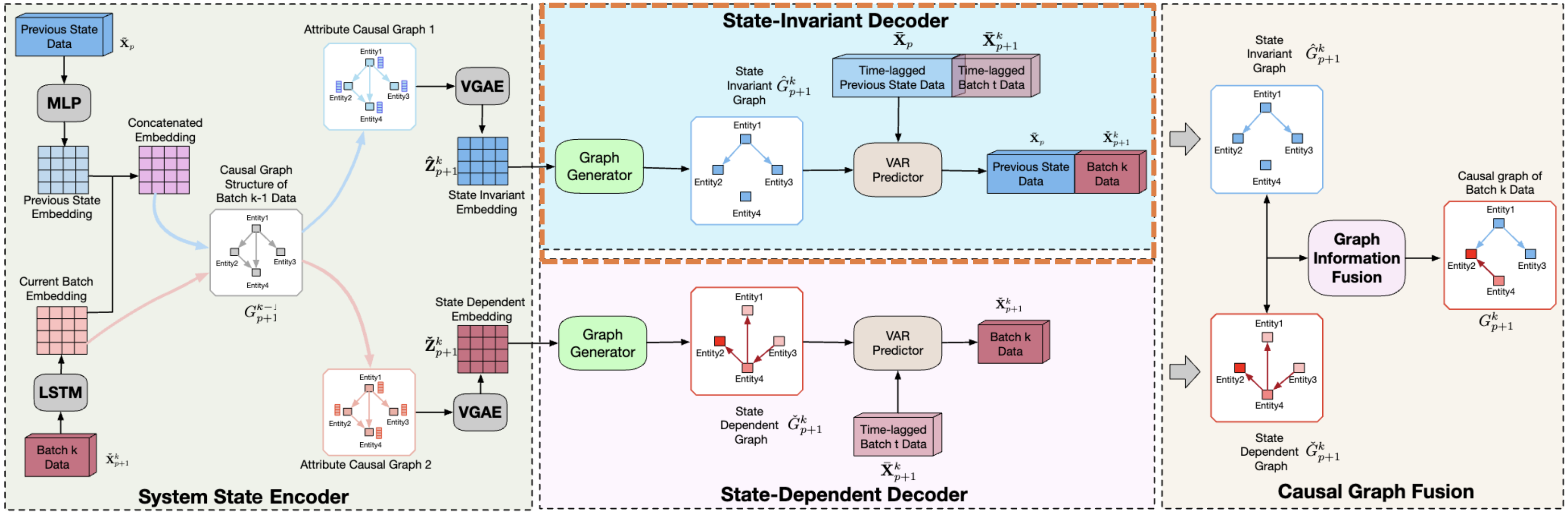
$$\mathbf{U}_p = \tilde{\mathbf{X}}_p \cdot \mathbf{W}_p + \mathbf{b}_p \qquad \mathbf{H}^k_{p+1} = f(\check{\mathbf{X}}^k_{p+1}, \mathbf{H}^{k-1}_{p+1})$$

- Map the learned embedding as **the attributes of the previous causal graph** and disentangle the information of the attributed causal graph by **VGAE**

$$\hat{\mathbf{Z}}^k_{p+1} = g(\mathbf{A}^{k-1}_{p+1}, \mathrm{Concat}(\mathbf{U}_p, \mathbf{H}^k_{p+1})) \qquad \check{\mathbf{Z}}^k_{p+1} = g(\mathbf{A}^{k-1}_{p+1}, \mathbf{H}^k_{p+1})$$

System state encoder disentangles the information of both data and causal graph

Disentangled Causal Graph Learning



State-Invariant Decoder

- Decode the state-invariant causal graph by **minimizing the error of reconstructed causal graph and the previous causal state causal graph**

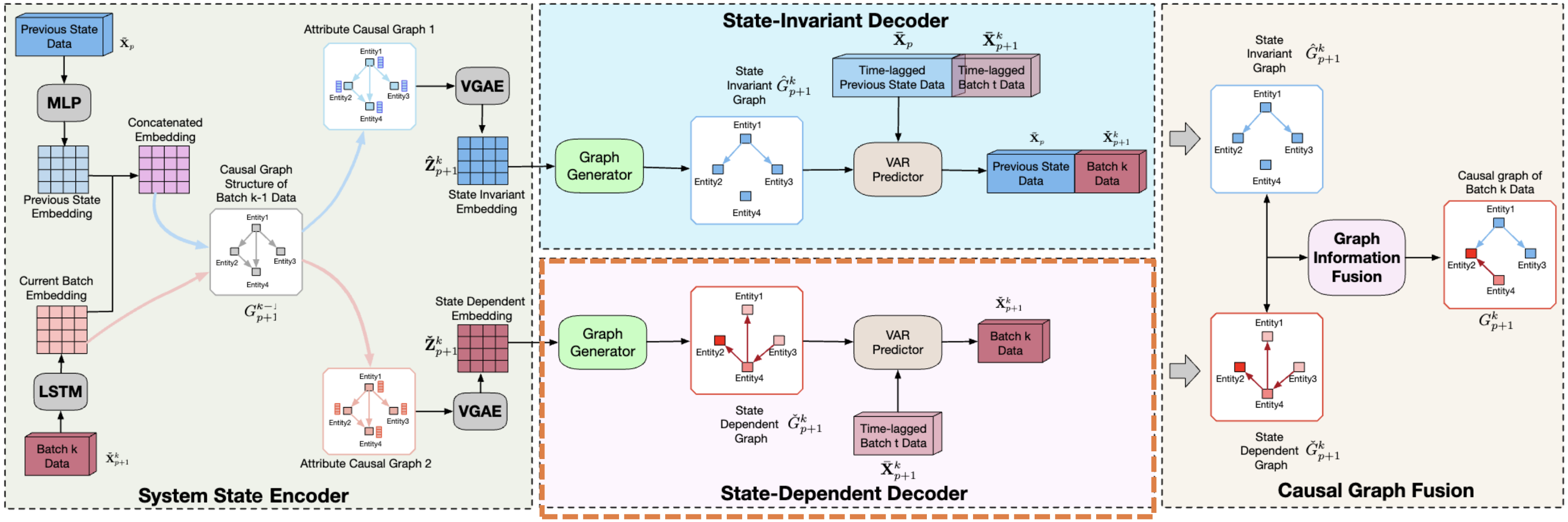
$$\hat{G}^{k}_{p+1} = \mathrm{Sigmoid}(\hat{\mathrm{Z}}^{k}_{p+1} \cdot \hat{\mathrm{Z}}^{k\top}_{p+1}) \qquad \mathcal{L}_{\hat{G}} = \left\| \hat{\mathrm{A}}^{k}_{p+1} - \mathrm{A}^{k-1}_{p+1} \right\|^2$$

- Rectify the learned state-invariant causal graph by **minimizing the prediction error on previous state data and current batch data**

$$\mathcal{L}_{\tilde{p}} = \left\| \tilde{\mathrm{X}}_p - (\tilde{\mathrm{X}}_p \cdot \hat{\mathrm{A}}^{k}_{p+1} + \bar{\mathrm{X}}_p \cdot \hat{\mathrm{D}}^{k}_{p+1}) \right\|^2 \qquad \mathcal{L}_{\hat{p}} = \left\| \check{\mathrm{X}}^{k}_{p+1} - (\check{\mathrm{X}}^{k}_{p+1} \cdot \hat{\mathrm{A}}^{k}_{p+1} + \bar{\mathrm{X}}^{k}_{p+1} \cdot \hat{\mathrm{D}}^{k}_{p+1}) \right\|^2$$

State-invariant decoder extracts invariant causation from the prior causal graph

Disentangled Causal Graph Learning



State-Dependent Decoder

- Decode the state-dependent causal graph by minimizing the error of reconstructed causal graph and **the complement** of the previous causal state causal graph

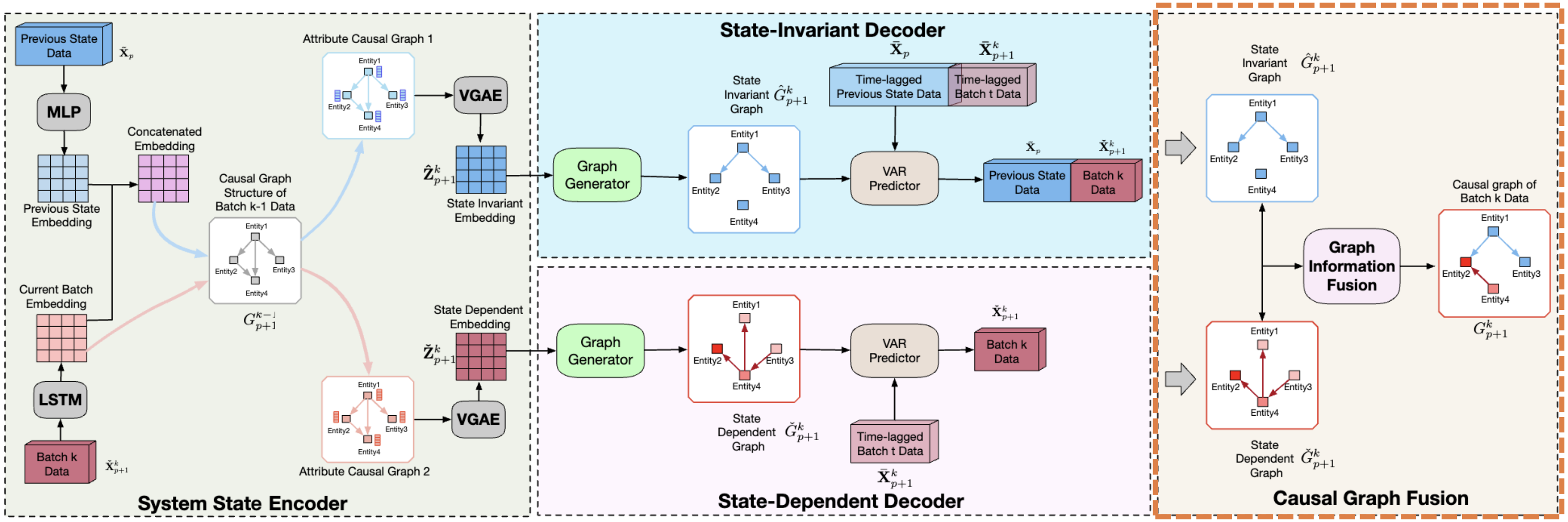
$$\breve{G}^k_{p+1} = \mathrm{Sigmoid}(\breve{\mathbf{Z}}^k_{p+1} \cdot \breve{\mathbf{Z}}^{k^\top}_{p+1}) \qquad \mathcal{L}_{\breve{G}} = \left\|\breve{\mathbf{A}}^k_{p+1} - (\sim \mathbf{A}^{k-1}_{p+1})\right\|^2$$

- Rectify the learned state-invariant causal graph by minimizing the prediction error on **current batch data**

$$\mathcal{L}_{\breve{p}} = \left\|\breve{\mathbf{X}}^k_{p+1} - (\breve{\mathbf{X}}^k_{p+1} \cdot \breve{\mathbf{A}}^k_{p+1} + \breve{\mathbf{X}}^k_{p+1} \cdot \breve{\mathbf{D}}^k_{p+1})\right\|^2$$

State-dependent decoder captures new causation from the graph's complement

Disentangled Causal Graph Learning



Causal Graph Fusion

- Remain the sparsity of the learned causal graph (Fusion Layer)

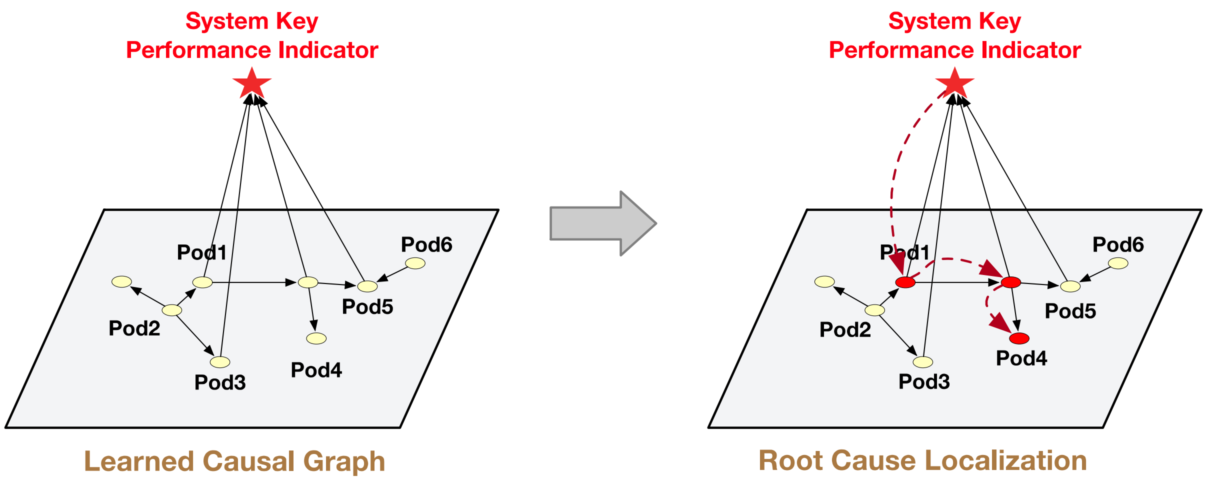
$$\mathrm{A}_{p+1}^{k} = \mathrm{RELU}(\tanh(\hat{\mathrm{A}}_{p+1}^{k} \cdot \check{\mathrm{A}}_{p+1}^{k\top} - \check{\mathrm{A}}_{p+1}^{k} \cdot \hat{\mathrm{A}}_{p+1}^{k\top}))$$

- Regularize the learned causal graph to be a directed acyclic graph (DAG), NOTEARS constraints

$$h(\mathrm{A}_{p+1}^{k}) = tr(e^{\mathrm{A}_{p+1}^{k} \circ \mathrm{A}_{p+1}^{k}}) - M$$

Causal graph fusion integrates both invariant and dependent causation

Network Propagation based RCA



- Random-Walk based RCA method, starting from the system KPI node and visits each node in the learned graph with restarts.
- The moving probability in the learned causal graph from node i to node j:

$$\mathbf{H}[i,j] = (1-\phi)\mathbf{A}^\top[i,j] / \sum_{\kappa=1}^{M} \mathbf{A}^\top[i,\kappa]$$

- The visiting probability transition equation of the random walk with restarts:

$$\mathbf{q}_{\tau+1} = (1-\varphi)\cdot \mathbf{q}_\tau + \varphi \cdot \mathbf{q}_\xi$$

- When the visiting probability distribution converges, the nodes' probability scores serve as their causal scores for ranking.

Network propagation RCA captures the propagation patterns of system failures

Framework Convergence Conditions

Two Convergence Conditions

- Learned causal graph converges, which is determined by the similarity of the edge distributions between two iterations

$$\varsigma_G = 1 - \mathrm{JS}(P(G_{p+1}^{K-1})||P(G_{p+1}^{K}))$$

- Learned root cause list converges, which is determined by the similarity of the detected root cause list between two iterations

$$\varsigma_l = RBO(\mathbf{l}_{p+1}^{K-1}, \mathbf{l}_{p+1}^{K})$$

- Integrate the two kinds of similarities

$$\varsigma = \alpha \cdot \varsigma_G + (1 - \alpha) \cdot \varsigma_l$$

Experimental Evaluation

Baselines

- PC
  - is a classic constraint-based method, which first decides skeleton, then directions.
- C-LSTM
  - captures the nonlinear Granger causality by using LSTM neural networks.
- Dynotears
  - is a score-based method that uses SVAR to construct dynamic Bayesian networks.
- GOLEM
  - employs a likelihood-based score function to relax hard DAG constraints in NOTEARS.
- NOTEARS
  - forms the structure learning problem as a continuous constrained optimization task.
- NOTEARS*
  - online-version of NOTEARS.
- GOLEM*
  - online-version of GOLEM.

Datasets

- Swat
  - 6 high-level nodes, 51 low-level nodes, 16 faults
- WADI
  - 3 high-level nodes, 23 low-level nodes, 15 faults
- AIOps
  - 5 high-level nodes, 234 low-level nodes, 5 faults

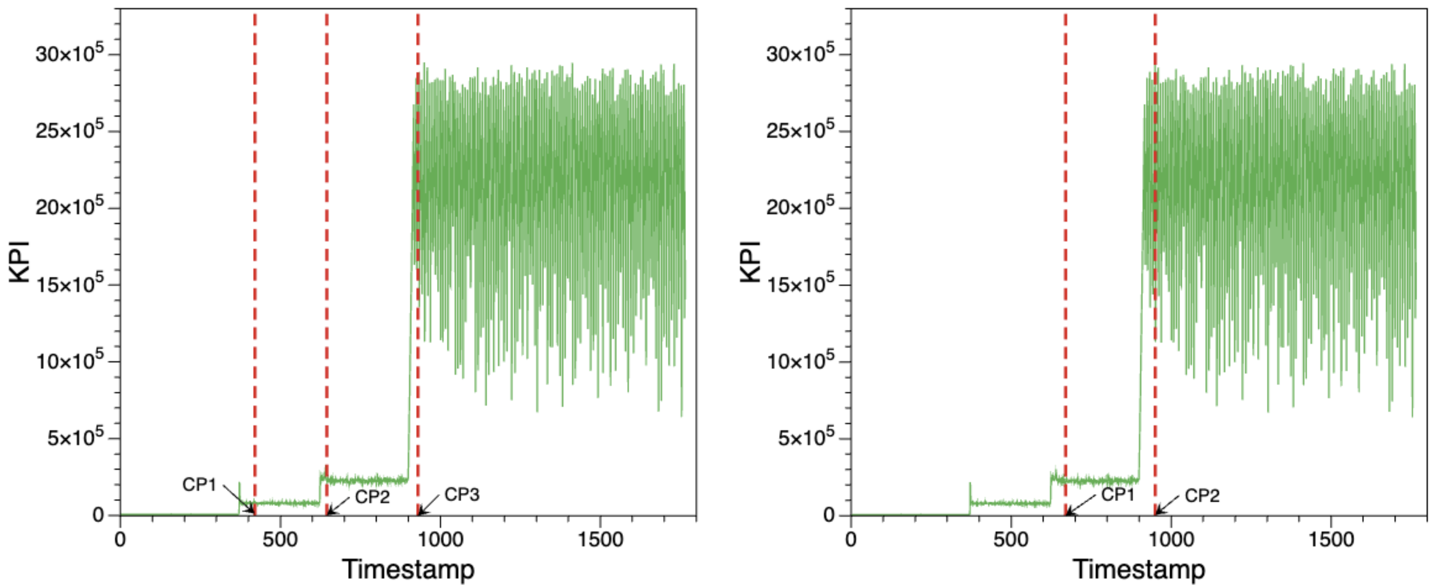
Evaluation Metrics

$$\mathrm{PR@K} = \frac{1}{|\mathbb{A}|} \sum_{a \in \mathbb{A}} \frac{\sum_{i<K} R_a(i) \in V_a}{min(K, |V_a|)},$$

$$\mathrm{MAP@K} = \frac{1}{K|\mathbb{A}|} \sum_{a \in \mathbb{A}} \sum_{1 \le j \le K} \mathrm{PR@j},$$

$$\mathrm{MRR} = \frac{1}{\mathbb{A}} \sum_{a \in \mathbb{A}} \frac{1}{rank_{R_a}}, \quad \mathrm{RP} = (1 - \frac{rank_{R_a}}{N}) \times 100\%,$$

Online Trigger Point Detection



(a) AIOps 0524 (Metric data + KPI) (b) AIOps 0524 (KPI)

Comparison of **trigger point detection with and w/o metric data.** Red dashed lines indicate the trigger points. Please note that the trigger points reflected on KPI data were actually detected based on system entity metrics data (200+ variables with 200,000+ timestamps).

- **Integrating metric data with KPI data enhances early and precise change point detection.**

CORAL successfully detects the system state change points

Overall Comparison

Table 1: Overall performance *w.r.t.* Swat dataset.

	PR@1	PR@3	PR@5	PR@7	PR@10	MAP@3	MAP@5	MAP@7	MAP@10	MRR
CORAL	6.25%	**31.25%**	**55.21%**	**64.58%**	**92.71%**	**15.63%**	**29.79%**	**39.73%**	**53.96%**	31.72%
NOTEARS	6.25%	7.29%	12.50%	39.58%	47.92%	7.64%	9.58%	16.96%	25.00%	22.36%
GOLEM	**18.75%**	7.29%	18.75%	54.17%	62.50%	11.81%	13.33%	22.02%	33.44%	30.42%
Dynotears	**18.75%**	25.00%	29.17%	41.67%	58.33%	23.96%	26.04%	29.17%	37.08%	**33.99%**
PC	12.50%	21.88%	36.46%	47.92%	53.13%	19.79%	26.04%	31.40%	37.40%	32.27%
C-LSTM	12.50%	27.08%	27.08%	39.58%	60.42%	19.44%	22.50%	26.49%	34.17%	32.86%
NOTEARS*	6.25%	29.17%	36.46%	55.21%	67.71%	14.93%	23.54%	32.59%	42.19%	26.30%
GOLEM*	6.25%	29.17%	42.71%	57.29%	68.75%	17.01%	26.04%	34.97%	43.65%	28.09%

Table 2: Overall performance *w.r.t.* WADI dataset.

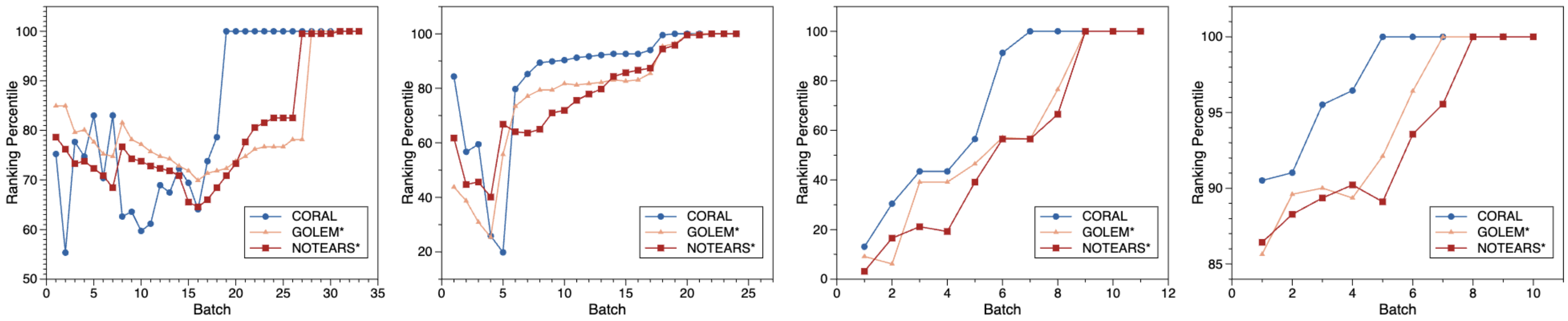
	PR@1	PR@3	PR@5	PR@7	PR@10	MAP@3	MAP@5	MAP@7	MAP@10	MRR
CORAL	**35.71%**	**23.81%**	**60.00%**	**70.24%**	**83.33%**	**28.71%**	**36.05%**	**45.82%**	**56.00%**	**51.90%**
NOTEARS	7.14%	23.81%	30.00%	35.71%	41.67%	17.46%	22.55%	26.14%	30.80%	30.12%
GOLEM	7.14%	10.71%	40.00%	51.19%	64.29%	9.52%	20.14%	28.33%	38.05%	25.89%
Dynotears	14.29%	29.76%	32.86%	42.86%	46.43%	20.63%	25.38%	29.35%	33.76%	34.28%
PC	14.29%	21.43%	35.71%	45.24%	57.14%	16.67%	24.29%	28.91%	35.71%	30.74%
C-LSTM	12.50%	21.43%	46.43%	52.38%	64.29%	17.86%	27.86%	34.69%	42.62%	33.28%
NOTEARS*	14.29%	20.24%	45.71%	66.67%	72.62%	18.65%	27.48%	38.67%	48.38%	37.74%
GOLEM*	21.43%	20.24%	60.00%	64.29%	73.81%	19.84%	30.33%	39.86%	48.98%	40.24%

Table 3: Overall performance *w.r.t.* AIOps dataset.

	PR@1	PR@3	PR@5	PR@7	PR@10	MAP@3	MAP@5	MAP@7	MAP@10	MRR
CORAL	**80.00%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**93.33%**	**96.00%**	**97.14%**	**98.00%**	**90.00%**
NOTEARS	0.00%	40.00%	80.00%	40.00%	60.00%	20.00%	28.00%	37.14%	44.00%	20.46%
GOLEM	20.00%	60.00%	60.00%	60.00%	60.00%	40.00%	40.00%	51.43%	54.00%	37.74%
Dynotears	40.00%	60.00%	60.00%	60.00%	60.00%	53.33%	56.00%	57.14%	58.00%	50.77%
PC	20.00%	20.00%	20.00%	40.00%	60.00%	20.00%	20.00%	22.86%	30.00%	25.36%
C-LSTM	0.00%	40.00%	60.00%	60.00%	60.00%	26.67%	36.00%	42.86%	48.00%	24.73%
NOTEARS*	40.00%	80.00%	80.00%	80.00%	80.00%	66.67%	72.00%	74.29%	76.00%	60.00%
GOLEM*	60.00%	80.00%	80.00%	80.00%	80.00%	73.33%	76.00%	77.14%	78.00%	70.00%

- CORAL significantly outperforms other baseline models in terms of all evaluation metrics.
- Online root cause analysis algorithms perform much better than other baselines.

Learning Procedure Analysis



(a) AIOps 0901 (b) AIOps 0524 (c) SWaT Fault 8 (d) WADI Fault 6

Comparison of online RCA models on different batches in terms of ranking percentile.

- **The performance of online RCA frameworks improves as the number of data batches increases.**
- **Online RCA successfully identifies root causes by gradually detecting changing patterns in monitoring metric data.**
- **CORAL achieves this advantage by updating the causal graph through disentangling state-invariant and state-dependent information, leading to more robust and effective causal structures.**

CORAL quickly and accurately locates the root cause of system failures

Conclusion and Future Work

- **Conclusion**
  - We propose an **incremental root cause analysis framework for mitigating damages and losses of system failures.**
  - **Online trigger point detection** module can detect system state changes **in a short delay time**.
  - Incremental causal discovery **disentangles system state-dependent and system state-invariant information** for efficiently updating causal model.
  - The proposed framework has been deployed in **real industrial systems** and plays an important role in keeping security.
- **Future Work**
  - We will extend our framework to other important real-world scenarios such as financial service, health care, and etc.

Thanks for Listening!