Prolego: Time-Series Analysis for Predicting Failures in Complex Systems

Anwesha Das[1,2] Alex Aiken[2]

[1]The University of Chicago [2]SLAC National Accelerator Laboratory/Stanford University

Autonomic Computing and Self-Organizing Systems (ACSOS)

29th September 2023

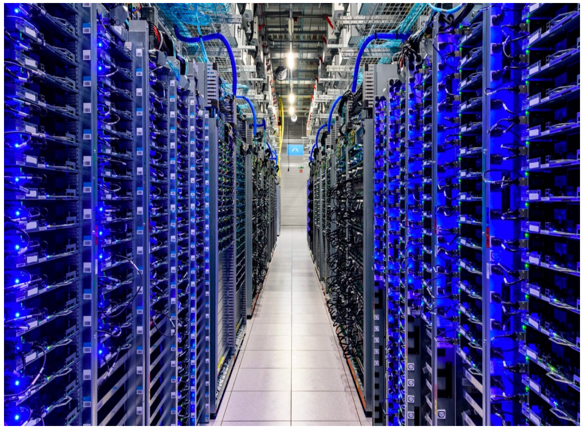
Stanford University

SLAC NATIONAL ACCELERATOR LABORATORY

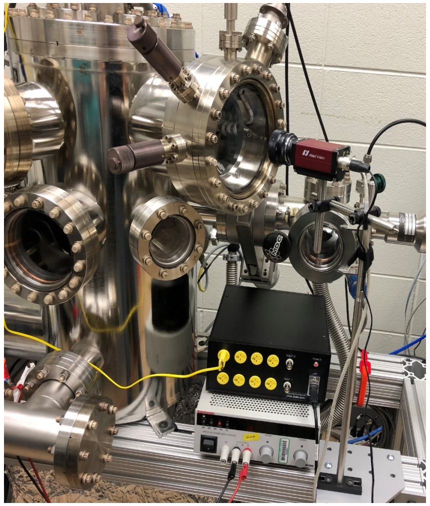
Large-Scale Systems Experience Failures

➔ Production Systems: Complex design, Heterogeneous components, Scale
  - Faults → Failures, Wasted Resources (computation, energy)
  - Fault Diagnosis and Recovery → Expensive (time, money)
  - Failures → Reduced system productivity, consume operator's time

Exascale Supercomputer



Unplanned Data Center Outages

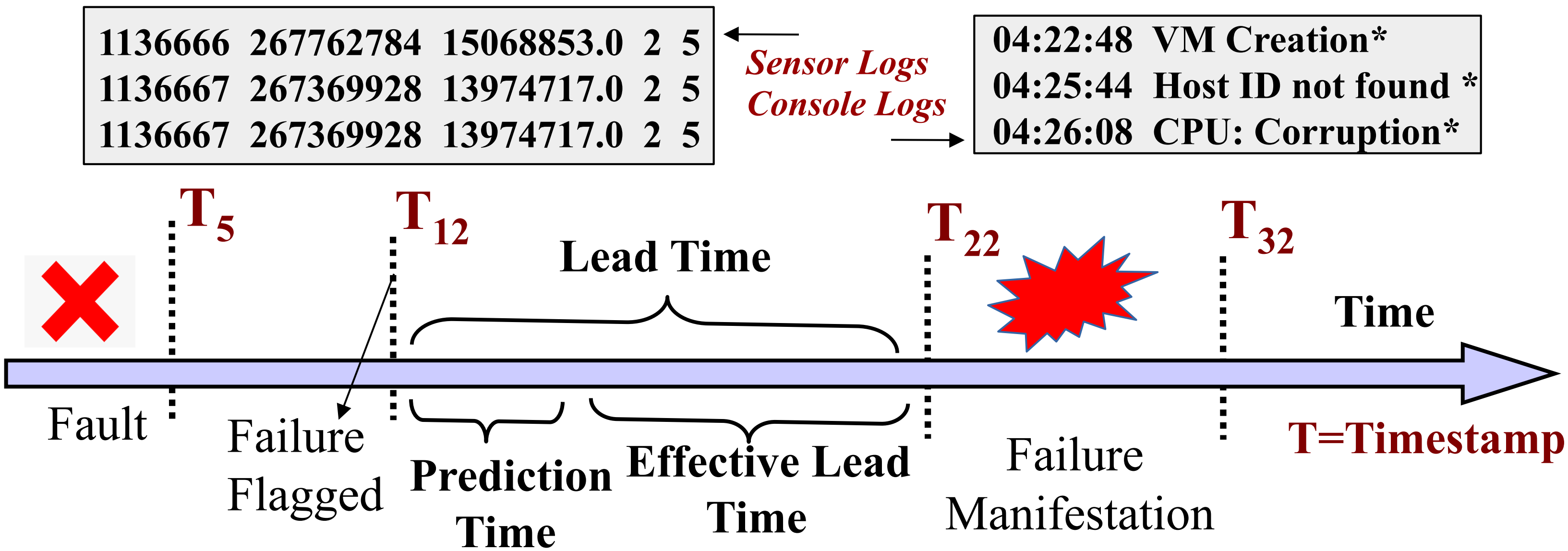


Integrated Devices in Cyberphysical Systems

What do most systems require? Efficient and accurate predictive maintenance !!

Failure Prediction

- ➔ Production Systems: Information-rich logs, Diverse log sources
  - Lead Time: Time left for failure to happen $\Delta(T_{22} - T_{12})$, failure flagged
  - Most failure studies → Lack lead time sensitivity study
  - Need? Unsupervised scalable approaches



- ➔ **Prolego**[1] → Failure prediction with multivariate time-series logs
- ➔ Can we reduce the time to predict?
  - Runtime support for lead time optimization

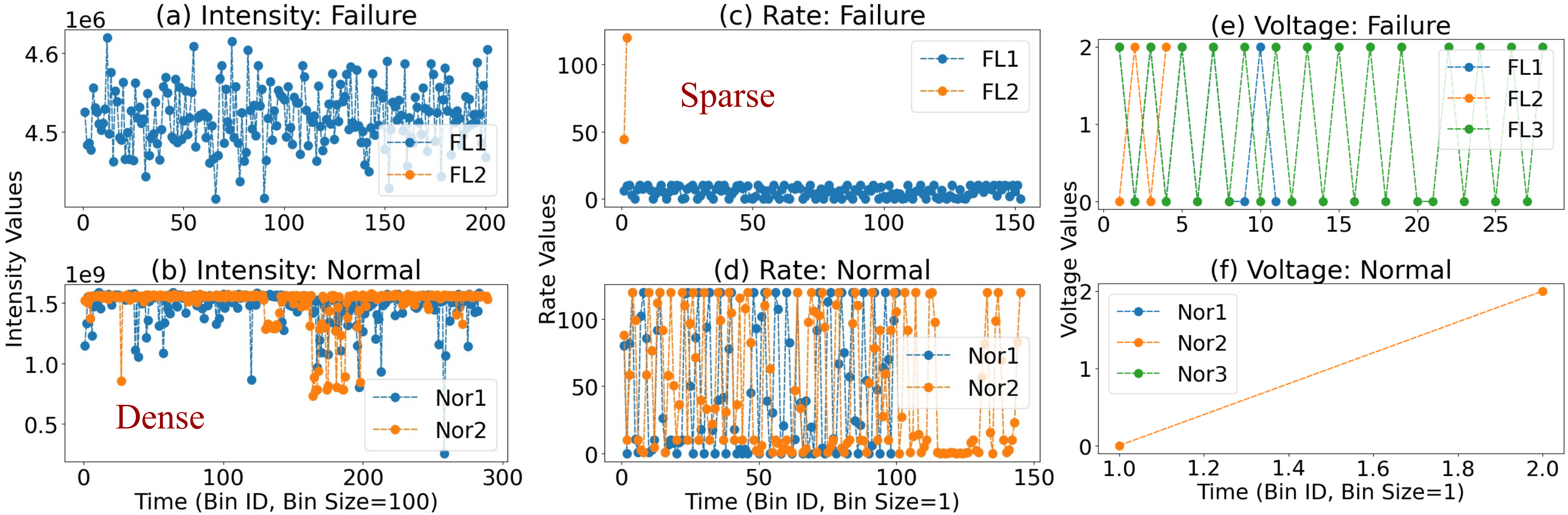
[1]Forecast or predict in Greek

Challenges

- **Sparse Ground Truth**
  - Lack of precise labels in the data
  - Manual Data Labeling: Inefficient, Cumbersome

- **Irregularities in Time-series**
  - Sparse and Dense; Skipped values for storage efficiency, Diverse sampling rates
  - Distinction between normal and faulty times ⟶ non-trivial

FL: Failure Nor: Normal



Failures: Less values, Lower magnitude, Missing values: Seen during normal times as well !!

Challenges

- **Unstable System States**
  - Application-specific configurations → influence the overall system state
  - Characteristics of normal behaviour change over time
    - Moderate/Low quality, limited training data

- **Diverse Failure Durations**
  - Few seconds to a few hours depending on the system and characteristic failures
  - Abrupt short-term failures → Not quite predictable

Approach

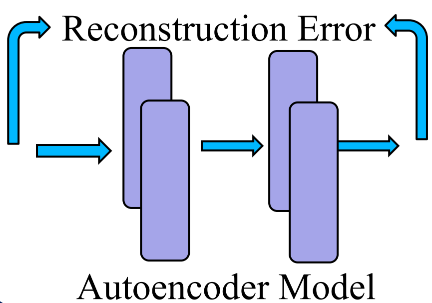
Signals

Approach

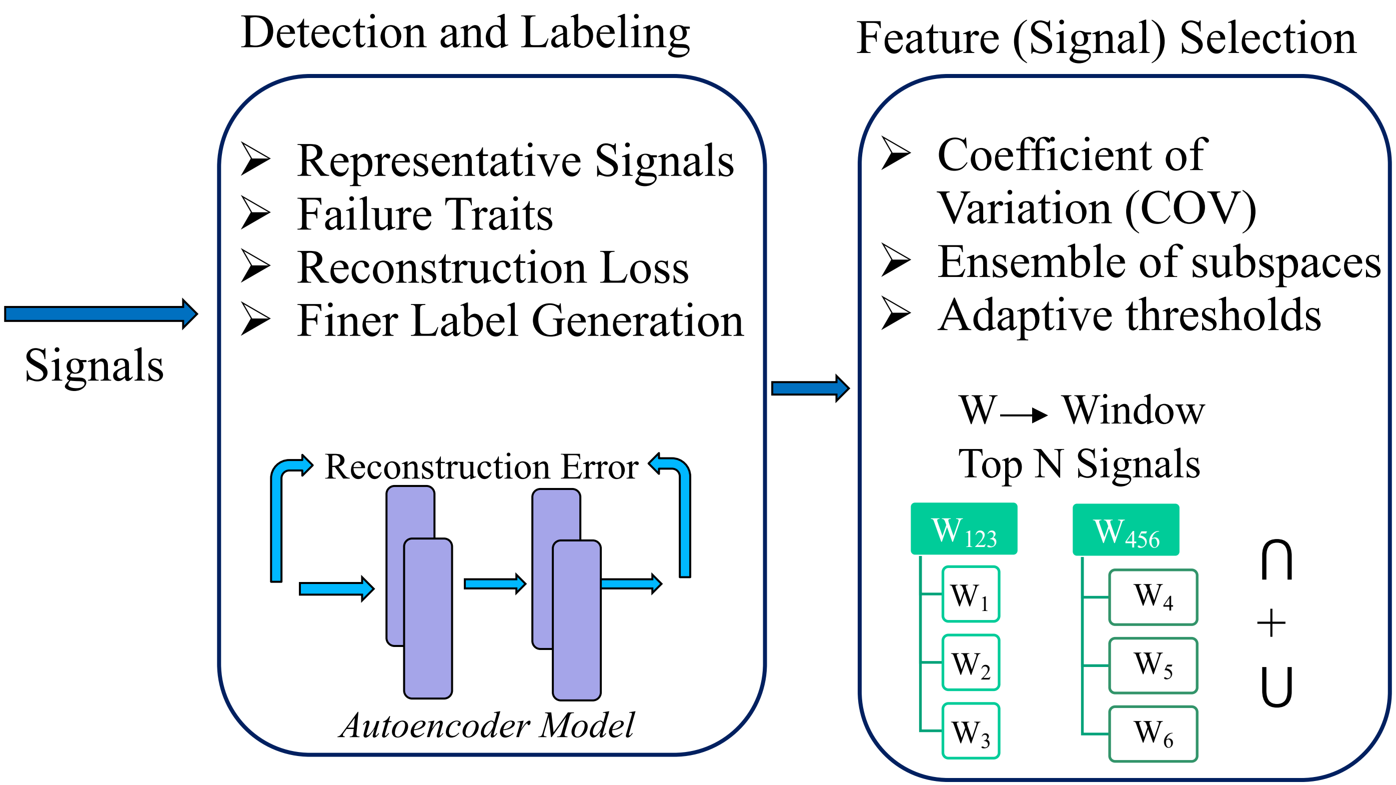
Detection and Labeling

- Representative Signals
- Failure Traits
- Reconstruction Loss
- Finer Label Generation

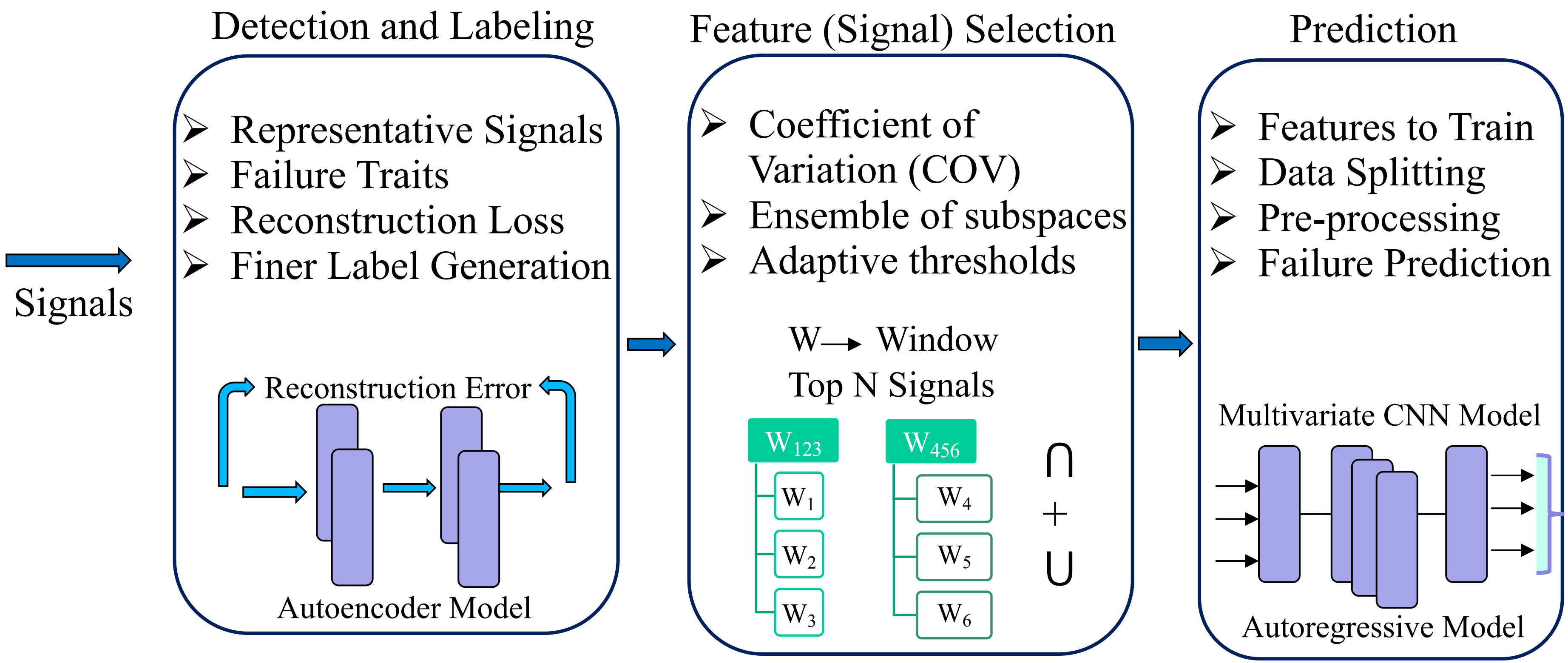
Signals



Approach

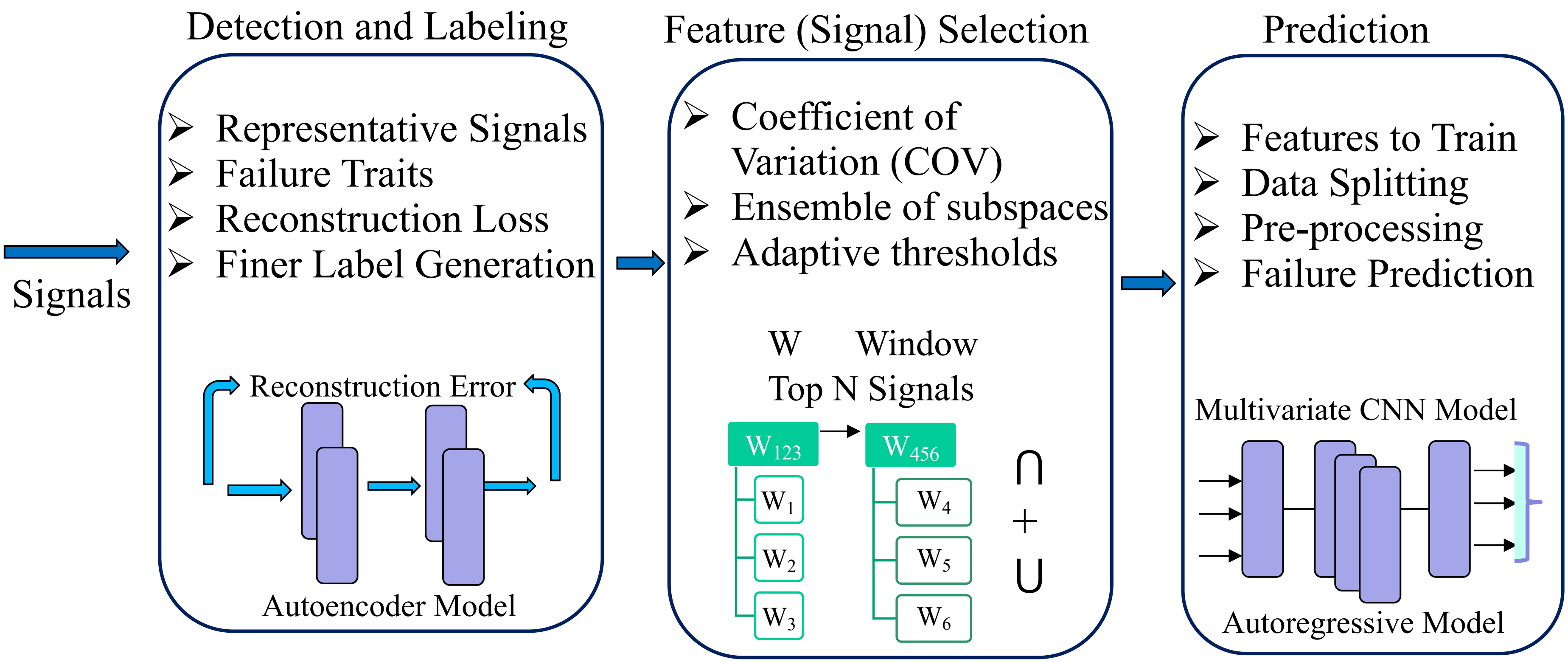

Detection and Labeling
Feature (Signal) Selection
Signals
Representative Signals
Failure Traits
Reconstruction Loss
Finer Label Generation
Reconstruction Error
Autoencoder Model
Coefficient of Variation (COV)
Ensemble of subspaces
Adaptive thresholds
W → Window
Top N Signals
W123
W1
W2
W3
W456
W4
W5
W6
∩
+
∪

Approach



Signals
Detection and Labeling
Representative Signals
Failure Traits
Reconstruction Loss
Finer Label Generation
Reconstruction Error
Autoencoder Model
Feature (Signal) Selection
Coefficient of Variation (COV)
Ensemble of subspaces
Adaptive thresholds
W → Window
Top N Signals
W_{123}
W_1
W_2
W_3
W_{456}
W_4
W_5
W_6
$\cap$
$+$
$\cup$
Prediction
Features to Train
Data Splitting
Pre-processing
Failure Prediction
Multivariate CNN Model
Autoregressive Model

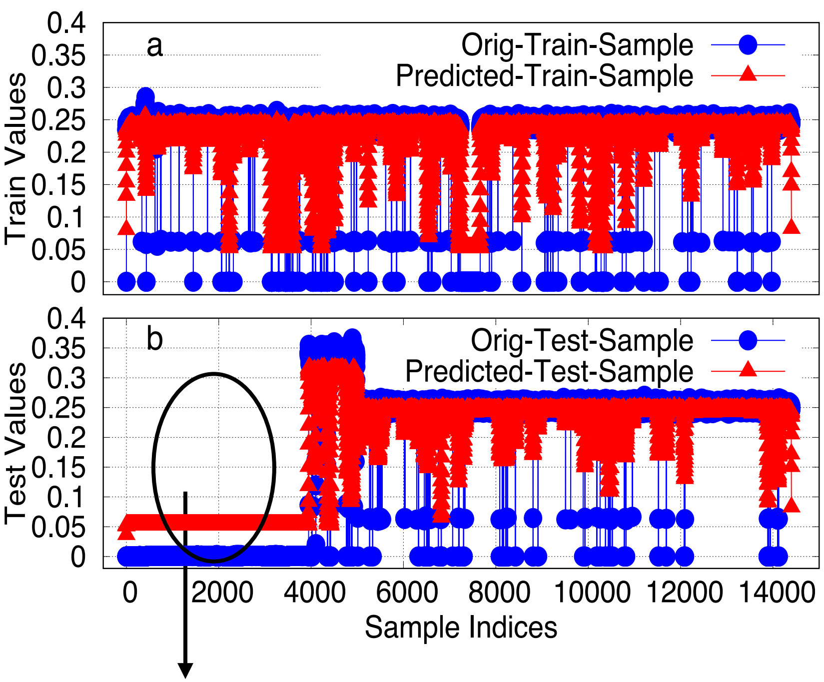
Approach



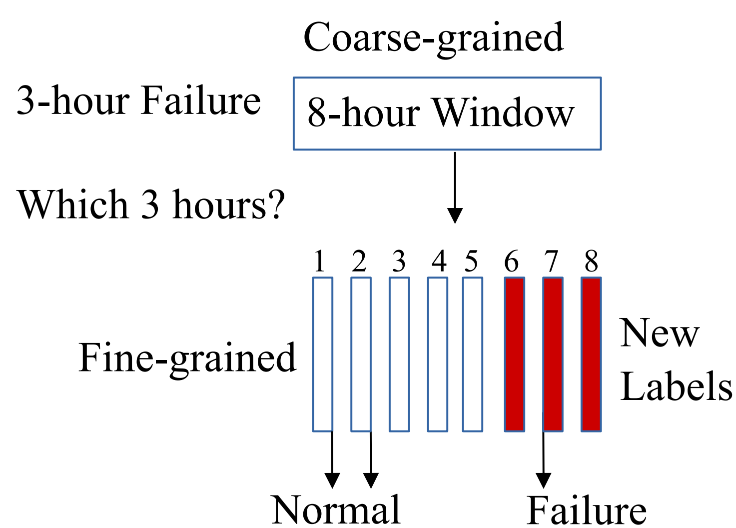
- **Prolego**: Three-phase design
  - Finer labeling → A few high-level performance-indicative signals used
  - Feature selection → With a larger parameter space
  - Prediction of an imminent failure → The shortlisted features are used
- Lead time optimization
  - Feasibility of runtime scalability → Leverage an existing programming system

Detection and Labeling

- Intuition: Statistical traits of failures (V_{fail}) → Incorporate in the ML model
- Identify few (1 or 2) signals that represent machine performance
- From coarse labels (sparse ground truth) ⟶ Separate failure vs. normal times
- Autoencoder Model → Estimate detection accuracy OR generate new labels for time-windows shorter than the known coarse labels



(V_{fail}=0) a) Train and b) Test Sample



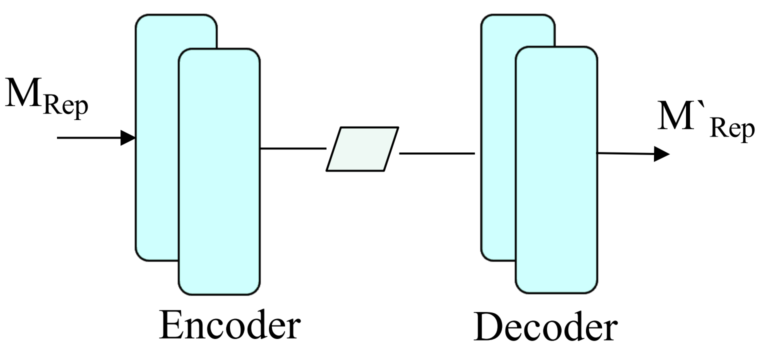
Detection & Labeling

A

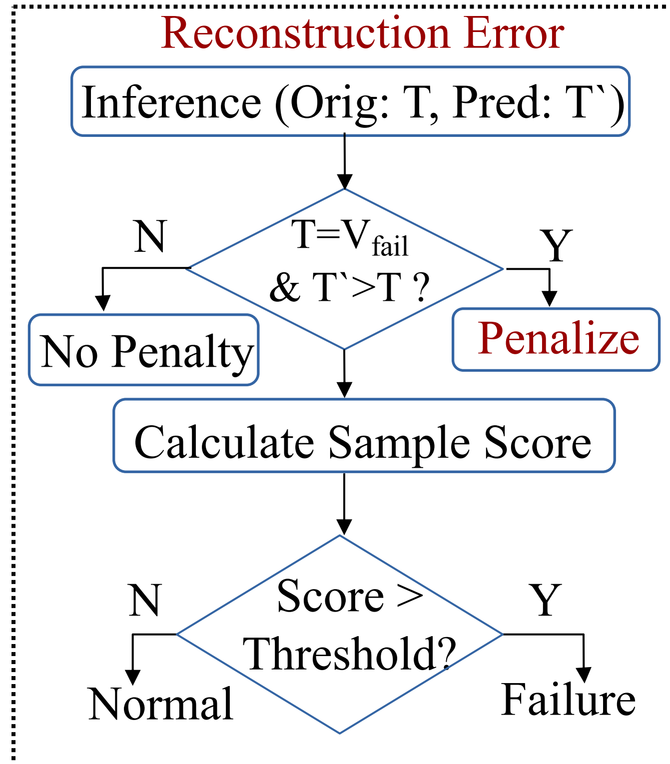
Detection and Labeling

- Intuition: Statistical traits of failures (V_{fail}) → Incorporate in the ML model
- Identify + Combine signal(s) to represent machine performance
- Coarse labels (sparse ground truth) ⟶ Separate failure and normal times
- Autoencoder Model[1] → Compute detection accuracy OR generate new labels for windows shorter than the known coarse-grained labels

Performance-Indicative Signal



Autoencoder Model



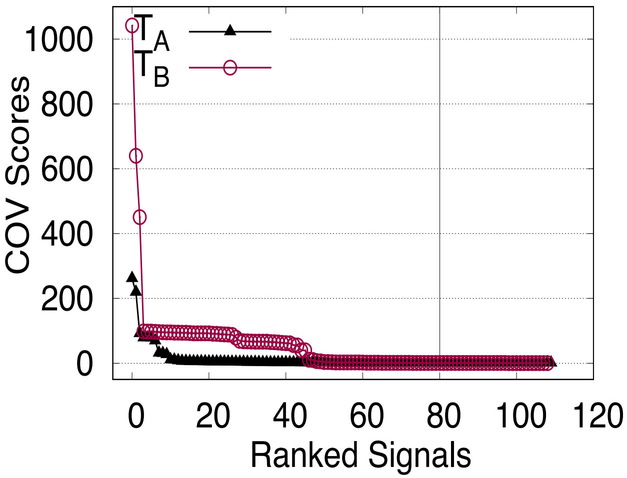
[1]Further details in the paper

Feature Selection

Signal Selection

B

- Intuition: *Select signals locally before forming an ensemble over multiple subspaces*
- Coefficient of variation (COV) scores (∂): Rank signals, bound the number of signals (threshold $\boldsymbol{\theta}$), Choose signals whose ($\partial > \boldsymbol{\theta}$)
- Feature selection across multiple time-windows
  - Common signals $\uplus$ Signals with ($\partial > \alpha$), α: derived from previous thresholds ($\boldsymbol{\theta}$) of shorter time-windows



After 1st 80 signals, score ~0.01

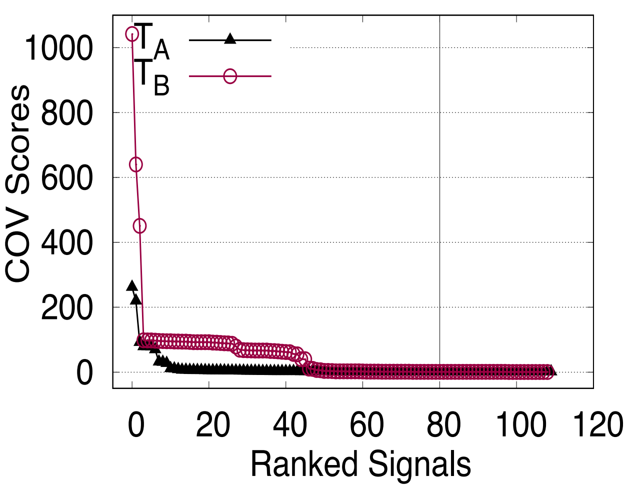
(T_A/T_B : Two time-windows)

Feature Selection

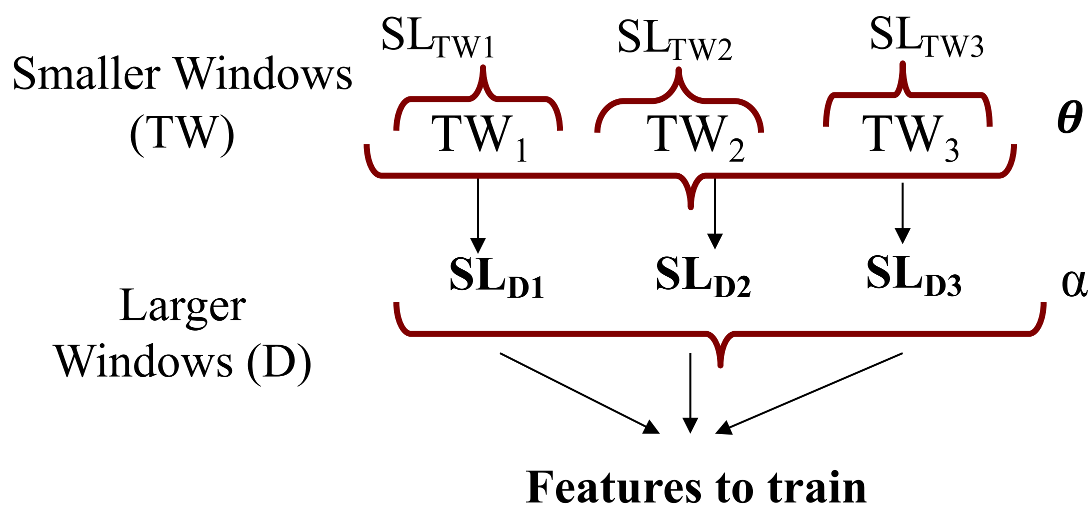
Signal Selection

B

- Intuition: *Select signals locally before forming an ensemble over multiple subspaces*
- Coefficient of variation (COV) scores (∂): Rank signals, bound the number of signals (threshold $\boldsymbol{\theta}$), Choose signals whose ($\partial > \boldsymbol{\theta}$)
- Feature selection across multiple time-windows
  - Common signals ⊎ Signals with ($\partial > \alpha$), α: derived from previous thresholds ($\boldsymbol{\theta}$) of shorter time-windows



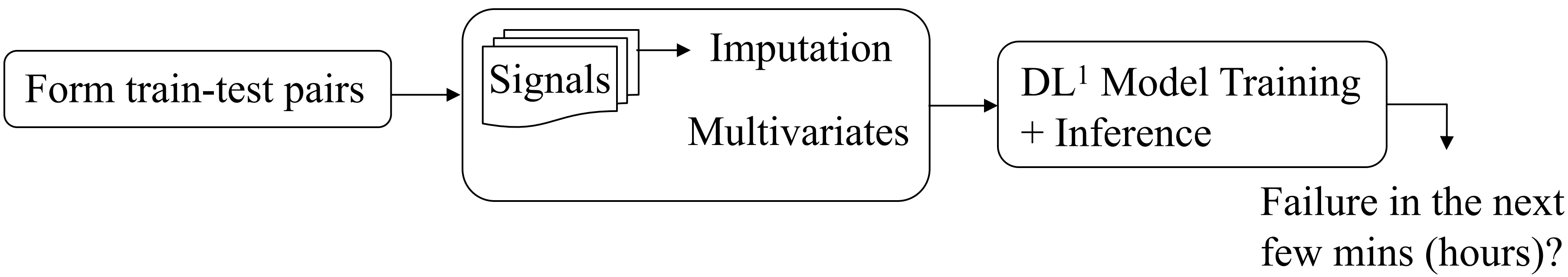
After 1st 80 signals, score ~0.01
(T_A/T_B : Two time-windows)



SL: Signal List

Prediction

- Data Splitting: Form train-test pairs (forward chaining if fewer training data)
- Across training windows: Signals to train based on COV-based feature selection
- Autoregressive Model: Dynamic Thresholding (γ), Forecast imminent failures



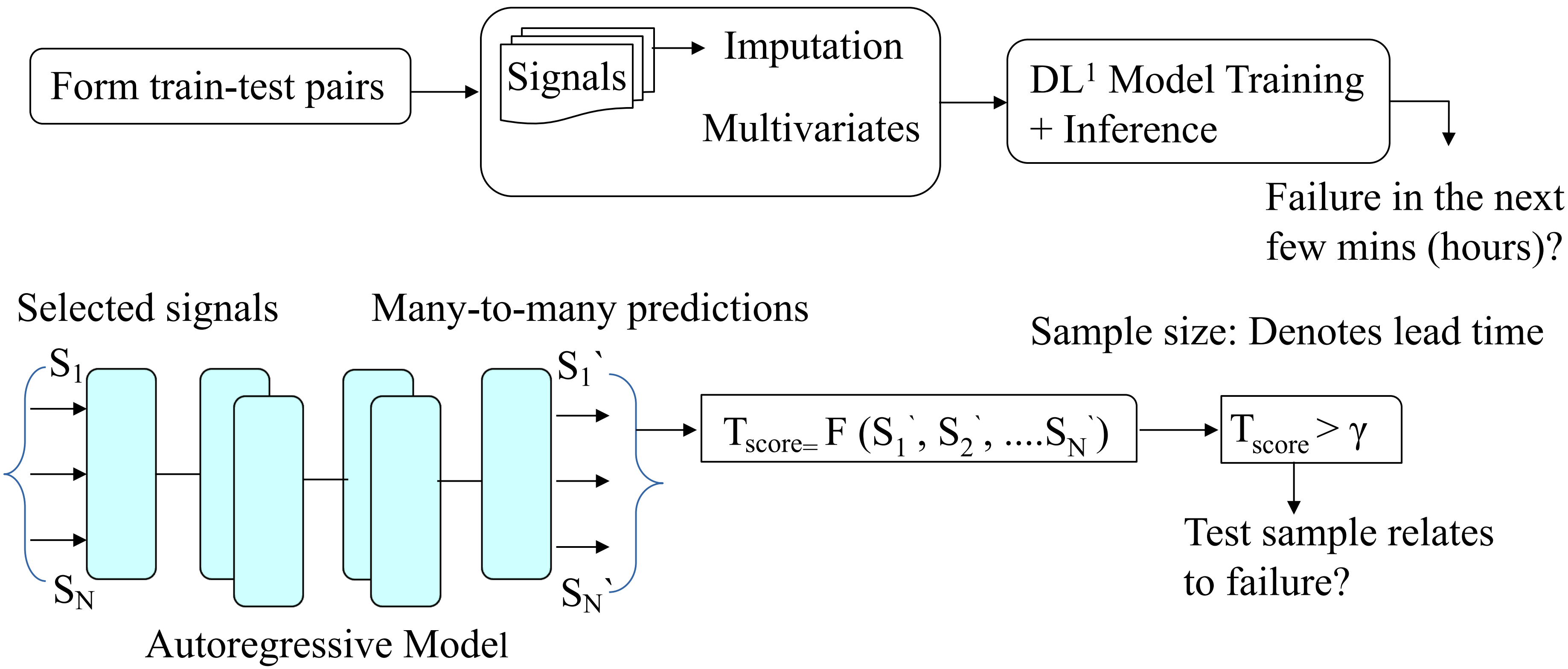
[1]Deep Learning

Prediction

Failure Prediction

C

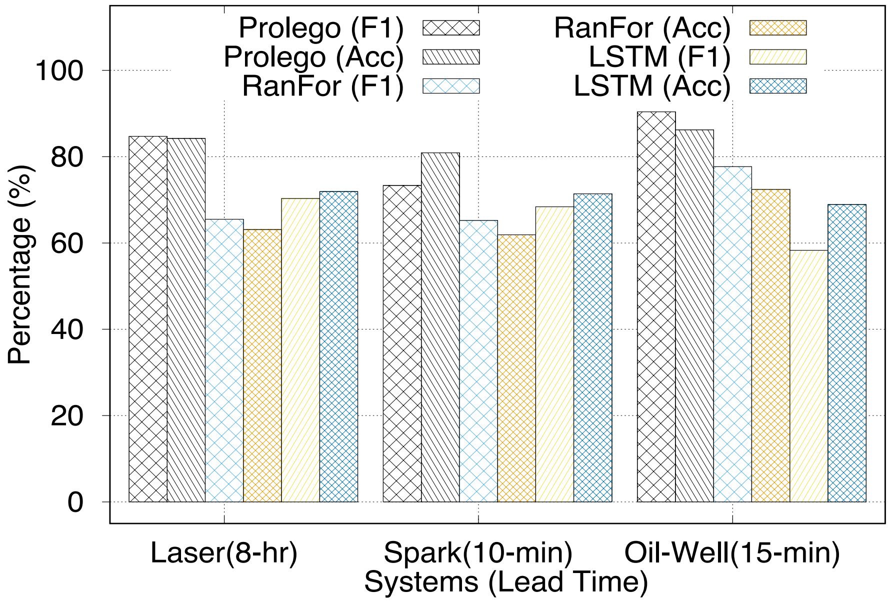
- Data Splitting: Form train-test pairs (forward chaining if fewer training data)
- Across training windows: Signals to train based on COV-based feature selection
- Autoregressive Model: Dynamic Thresholding (γ), Forecast imminent failures

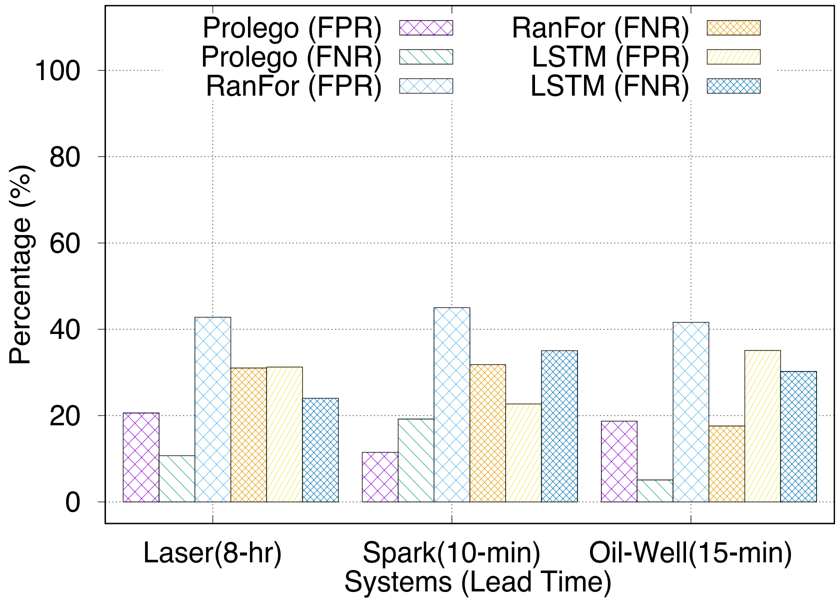


[1]Deep Learning

Results

- ❖ System Logs: X-ray Laser (LCLS[1]), Apache Spark Cluster, Oil Plant
  - ➢ Domains: High-Energy Physics, Distributed System, Petroleum Industry
- ❖ Baseline Comparisons: Random Forest (RanFor), Long Short-Term Memory (LSTM)

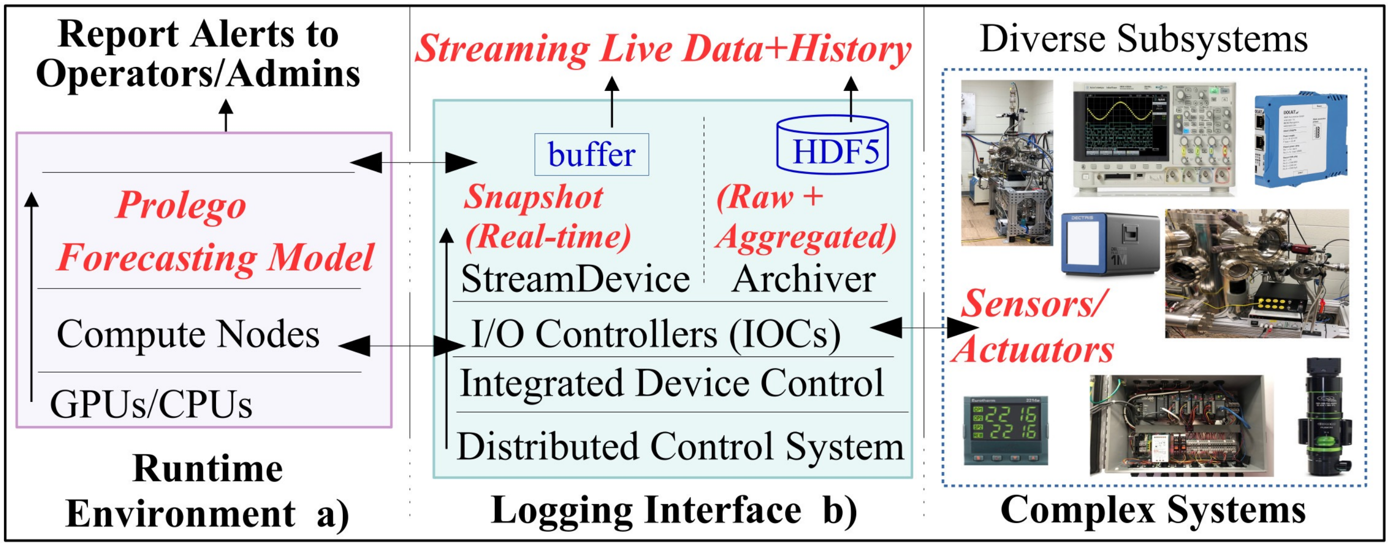




Prolego: > 80% F1 score and accuracy (Acc), false positive (FPR) and false negative rates (FNR) < 21%, with 5 mins to 8 hours of lead time !!

[1]Linac Coherence Light Source

Prolego Application

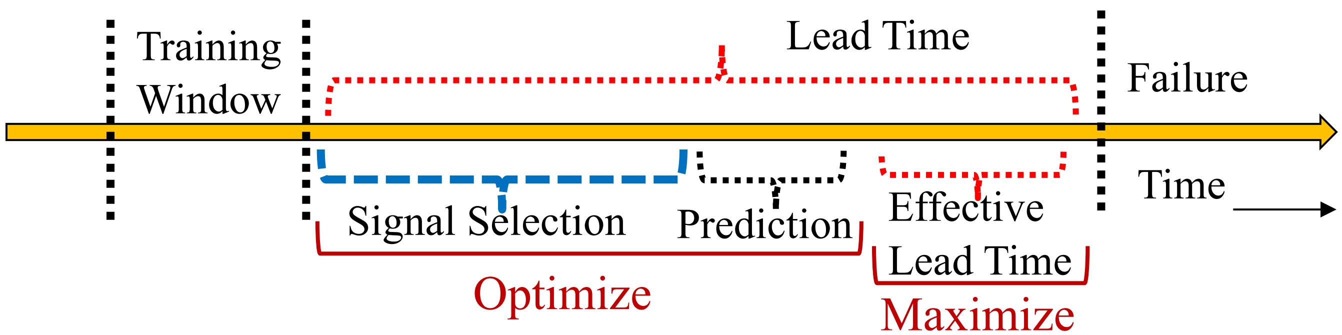


- Complex System Monitoring:
  - Larger scale, Increasing heterogeneity
  - Systems for ML, Programming Systems + Predictive Models
    - Legate + Prolego

Larger and complex systems ⟶ ***Harness the power of runtime systems and learning models***

Results

❖ Lead time optimization: Scalability in case of continuous forecasting
❖ Performance optimization with Legate[1] task-based programming model



❖ Apply Legate for Pandas mean and variance operation to compute COV
  ❖ Can the feature selection time be reduced ?

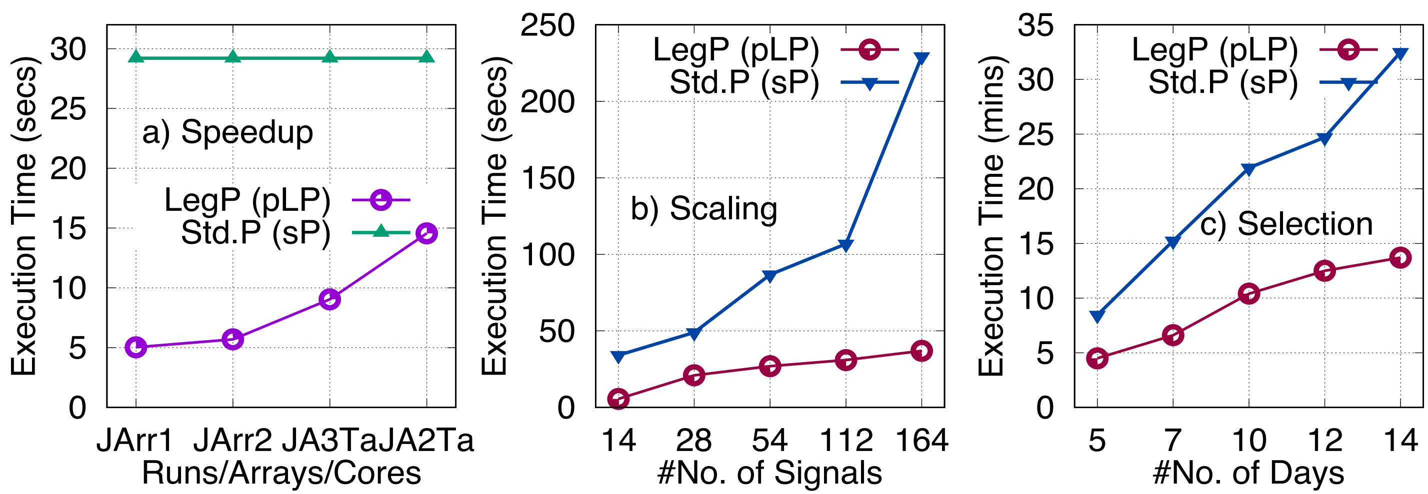
❖ Experiments: Different HPC and Legate parameters versus single node computation without legate, using ~164 signals (small scale)

Legate: [1]https://github.com/nv-legate/legate.pandas (Baur et al. SC'19)

Performance Optimization

- Fixed input size (14 signals) → LegP is 2x to 6x faster than StdP
- As signals increase (14 to 164) → LegP is 2x to 6x faster than StdP
- Increasing time-range (5 to 14 days) → LegP is 2x faster than StdP

LegP: Accelerated Pandas, StdP: Standard Pandas



Different Parallel Settings   Increasing #Signals   Ranking Time

With higher dimensions (e.g., $O(10^3)$ to $O(10^6)$ signals), predictive models and scalable runtime models together has the potential to enable timely system maintenance !!

Conclusion

- Prolego: Prediction of failures from multivariate sensor logs
  - Evaluation on three diverse systems
  - 5 mins to 8 hours of lead time
  - Over 80% prediction accuracy

- Demonstration of opportunities for lead time optimization
  - Using Legate programming system
  - Feature selection time → At least 2x faster
  - Prediction Model + Distributed Scalable Programming Model
    - Potential to enable timely system health monitoring

Code: https://github.com/adaptsyslearn/Prolego

Prolego forecasts failures on diverse systems with minimal expert supervision

Thank You