USENIX ATC '22

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

Primo: Practical Learning-Augmented Systems with Interpretable Models

Qinghao Hu[1], Harsha Nori[2], Peng Sun[3], Yonggang Wen[1], Tianwei Zhang[1]

1 Nanyang Technological University Singapore | S-Lab for Advanced Intelligence

2 Microsoft

3 商汤 sensetime

Machine Learning in Systems

- **Learning-Augmented System** is an emerging research topic

Storage

DeepSketch [*FAST '22*]: 33% Data Reduction

LinnOS [*OSDI '20*]: 80% Latency Reduction

HDDse [*ATC '20*]: 58x System Reliability

…

Cluster Scheduling

Sinan [*ASPLOS '21*]: 68% Resource Conservation

Helios [*SC '21*]: 6x JCT Reduction

FIRM [*OSDI '20*]: 16x SLO-Violation Reduction

…

Network

Clara [*SOSP '21*]: 89% Throughput Improvement

NeuroPlan [*SIGCOMM '21*]: 17% Cost Saving

LRB [*NSDI '20*]: 25% WAN Traffic Reduction

…

Security

FARE [*NDSS '21*]: 100% Fake Accounts Blocking

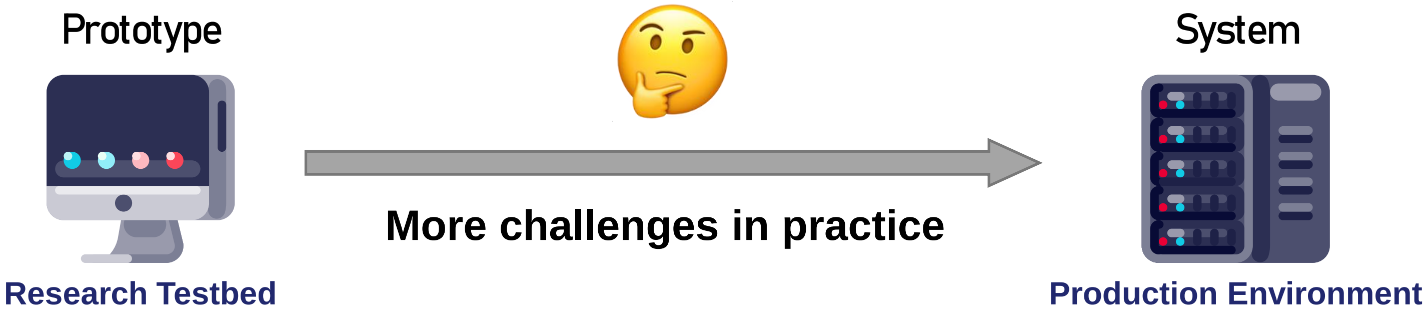
Apichecker [*EuroSys '20*]: 96% Malware App Recall

AdGraph [*S&P '20*]: 95% Accuracy in AD Blocking

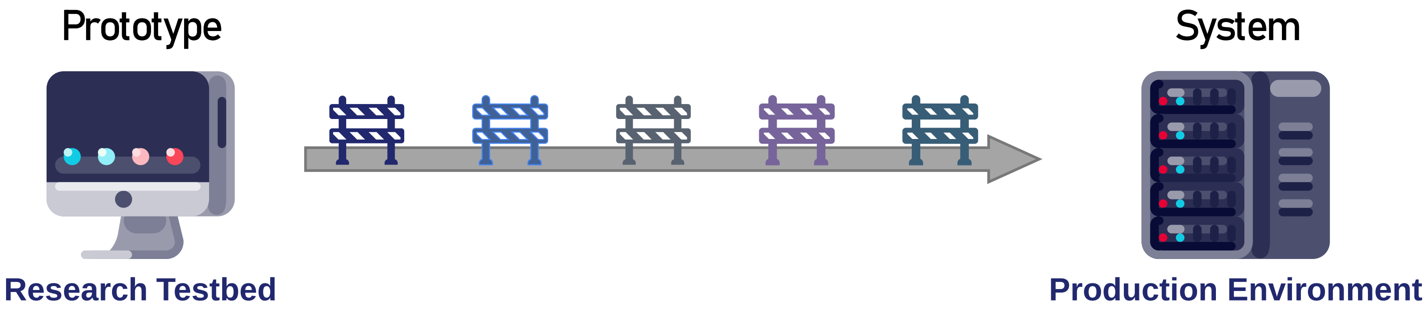
…

ML Brings Awesome System Improvement!

Challenges in Practical Deployment



Challenges in Practical Deployment



1 High Training and Tuning Cost

- **Continuous Model Fine-tuning / Retraining is Necessary**

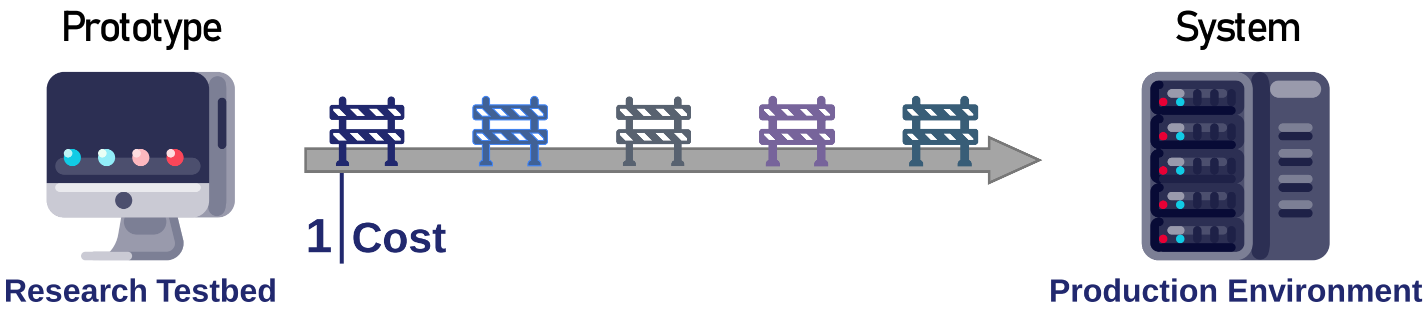
  1) System environment change: scale up ⮉ / down ⮋ over time

  2) Workload change: distribution drift

- **Microsoft Operation Experience** [*AutoSys*, *ATC ‘20*]

  1) Cost often exceed enterprise expectation

  2) Performance in testbed might not match the production environment



2 Strict Inference Overhead Requirement

Latency Constraint

AI Apps: ~10ms

ML-Sys: ~10us

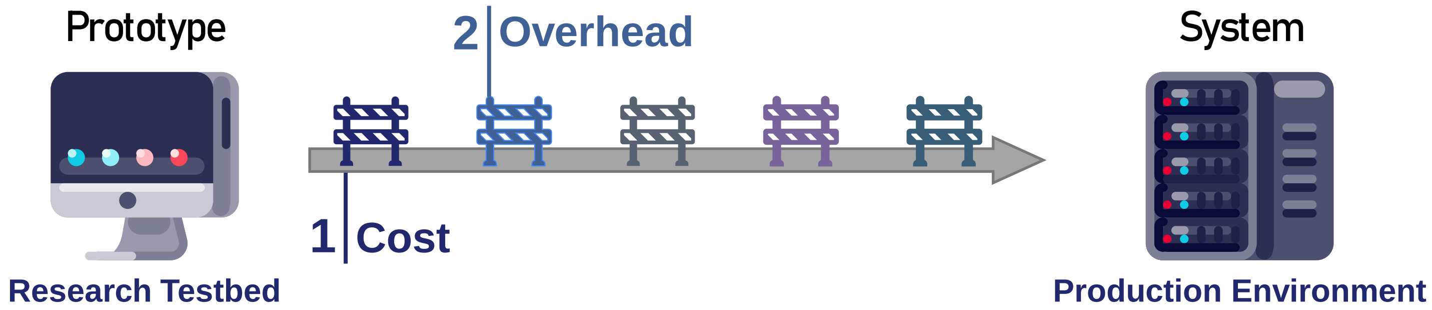
Resource Constraint

CPU

Memory

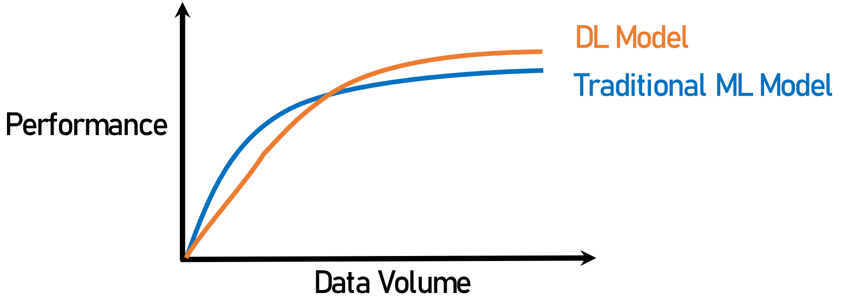
Storage

- **Limited Scalability** Testbed-scale ✓ Production-scale ✗
- **Side-effect to Production Workloads** ML models occupy too much resources



3 Insufficient Data

ML Model is Data-Driven



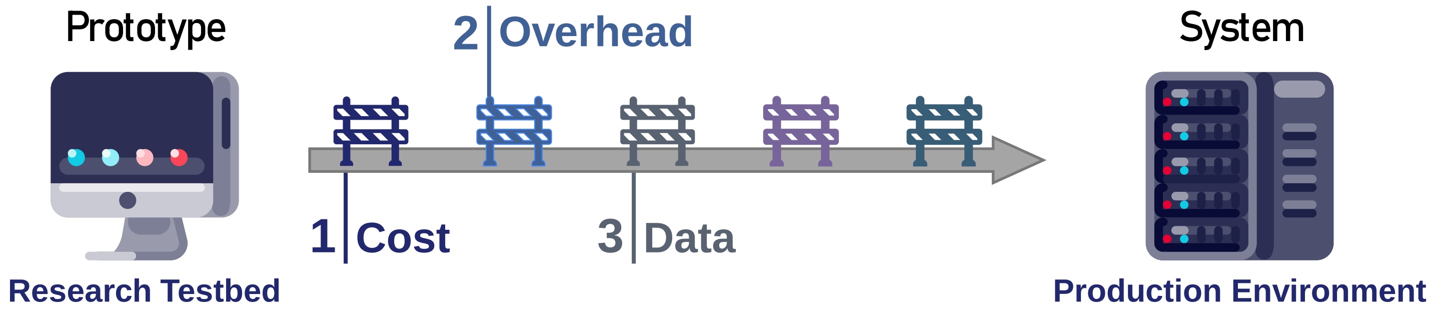
More complex model needs more data !

Some Scenarios Meet Data Issue

High Data Collection Cost   Sensitive / Privacy-related Data   ...

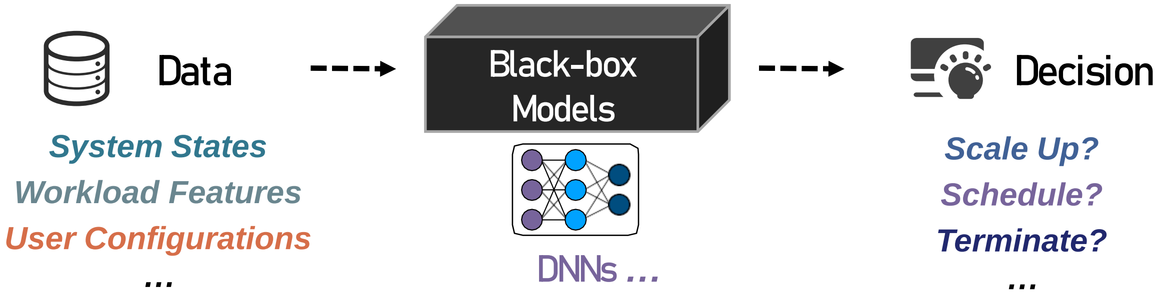
- **Data Augmentation and Synthesis Techniques**

  → Possible induce bias and distribution drift   → Not work in practice



4 Opaque Decision Process

Many Learning-Augmented Systems Rely on Black-Box Models

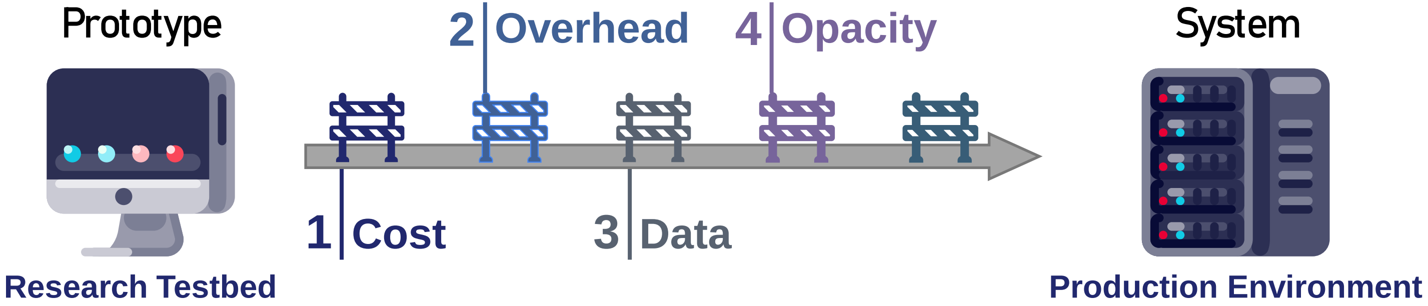


How the model make decision?

Should I trust the prediction?

- **Interpretability is Important But Often Ignored**

Operators need sufficient confidences to deploy learning-augmented systems

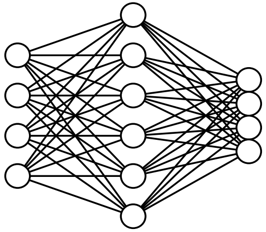


5 Hard to Adjust

System Adjustment is Needed

Different: System Scale Machine Type

Too Complex for System Operators



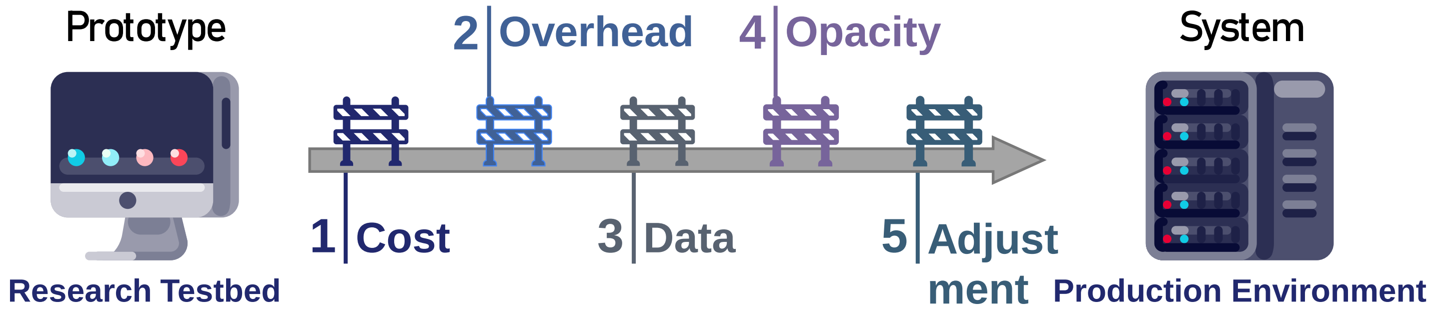
How to adjust the model?

Is the adjustment correct?

How to determine layers / neurons?

- **Improper Modifications → Severe Performance Degradation**

  Operators need guided or automatic model adjustment



1 | Cost

2 | Overhead

3 | Data

4 | Opacity

5 | Adjustment

How to address these issues?

Existing Solution

Interpreting Black-box Models

Security: LEMNA [*CCS '18*]

DeepAid [*CCS '21*]

Network: Metis [*SIGCOMM '20*]

Create another **surrogate model** to explain the **original model**

Limitations

	Interpretation			Training Cost	Inference Overhead	Insufficient Data
	Individual	Entire	Fidelity			
Interpreting Black-box	✔	✘	✘	✘	✘	✘

Any solution that can solve all challenges?

Our Approach

~~Interpreting Black-box Models~~

Adopt and Optimize Interpretable Models Directly

Linear Regression   Logistic Regression   Decision Tree   …

Benefits

Inherently Intelligible   Simple & Lightweight

	Interpretation			Training Cost	Inference Overhead	Insufficient Data
	Individual	Entire	Fidelity			
Interpreting Black-box	✓	✗	✗	✗	✗	✗
Interpretable Model	✓	✓	✓	✓	✓	✓

Why Interpretable Models Work

Major Concern --- Is there a trade-off between model accuracy and interpretability? 🤔

Key Observations

1. Input feature

AI Apps: Image Pixels Word Embeddings

ML-Sys: System States Workload Features

Meaningful and Lower Dimensional

2. Model Scale

AI Apps:	ResNet-18	11M	params
	BERT-Base	110M	params
ML-Sys:	RL-Sys	<10K	neurons[1]

Smaller Scale and Latency Sensitive

Interpretable Models have Comparable Performance and Less Overhead

[1]: whiRL [SIGCOMM '21]

Primo Design

Primo (Prior-based interpretable model optimization)

Objective --- Transparent, Accurate and Lightweight Learning-Augmented Systems

Different System Requirements

1 Online Systems

Real-time Response

Performance-Latency Trade-off

2 Offline Systems

No Latency Requirement

Focus on Performance

Primo support various interpretable models

Main Modules

Interpretable Models Training

Post-Processing Optimization

Interpretable Models Training

- Two Interpretable Models

PrAM: Addictive Model based Method

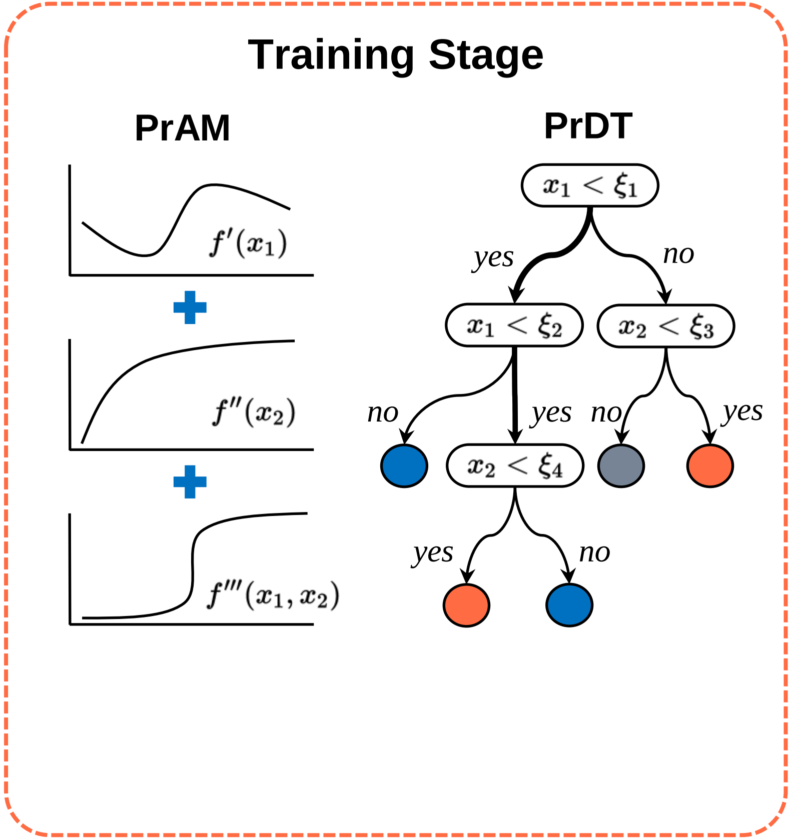
Summation of univariate or bivariant shape functions

Purpose: For better prediction accuracy

PrDT: Decision Tree based Method

Each decision can be clear visualized

Purpose: For strict latency and computation resource



Interpretable Models Training

- **Bayes Optimization**

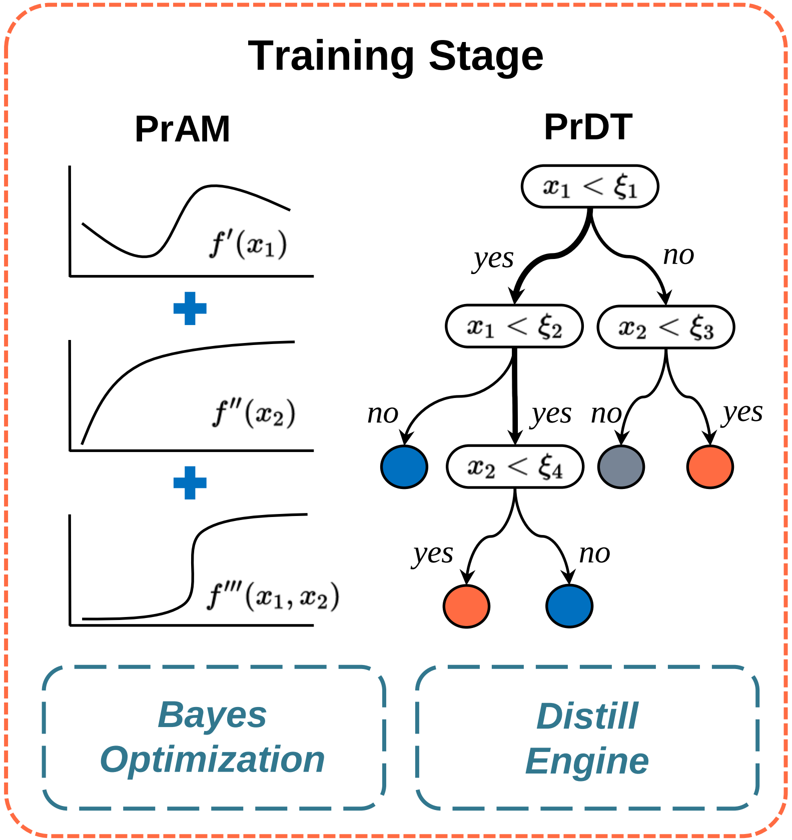
  Efficient search for the optimal model configuration

  Purpose: For accurate and succinct model

- **Distill Engine**

  Mimic the behavior of the original model

  Purpose: For RL-based system support



Post-Processing Optimization

- Not Necessary Step
- Two Post-Processing Tools

Monotonic Constraint

Edit shape functions according to prior knowledge

Purpose: For automatic model adjustment

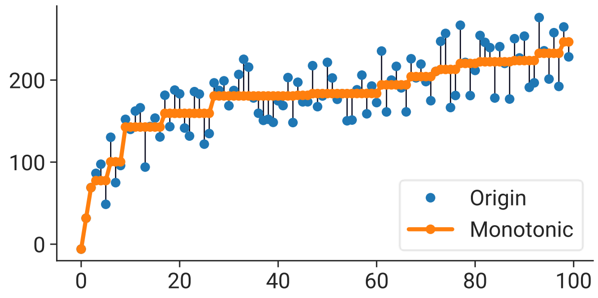
Counterfactual Explanation

Find smaller feature value change

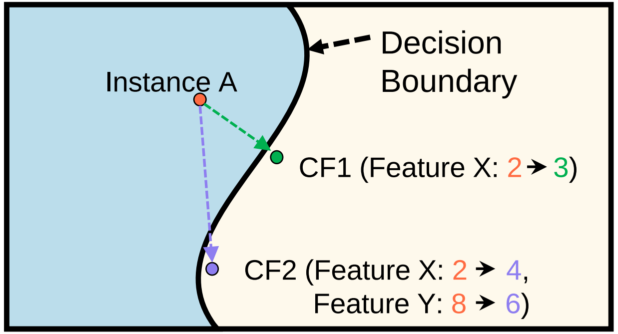
Purpose: For guided model adjustment

Post-Processing Stage

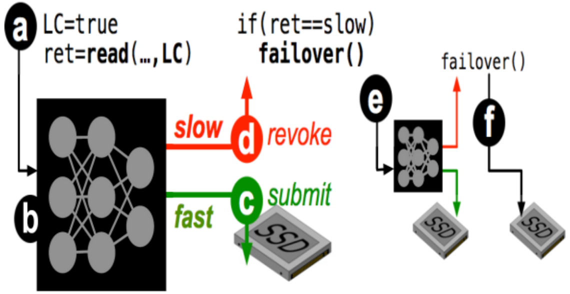
Monotonic Constraint



Counterfactual Explanation



Case Studies

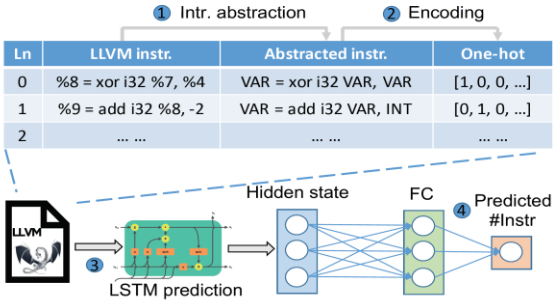


LinnOS *[OSDI '20]*

Flash Storage I/O

DNN

Online

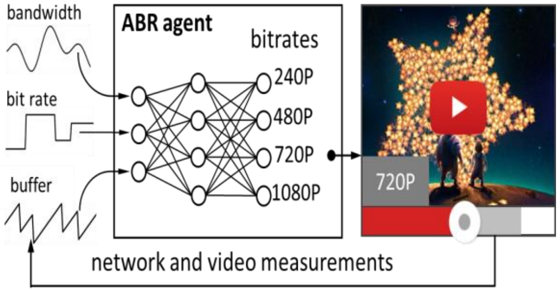


Clara *[SOSP '21]*

SmartNIC Offloading

LSTM, GBDT, SVM

Offline



Pensieve *[SIGCOMM '17]*

Video Streaming

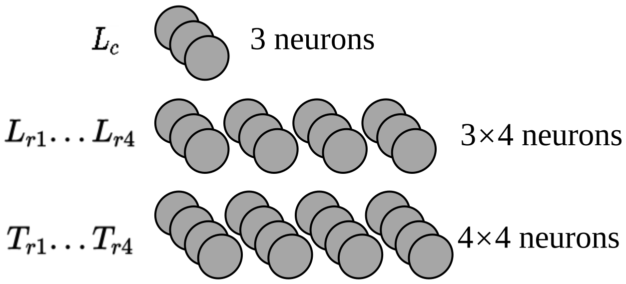
RL

Online

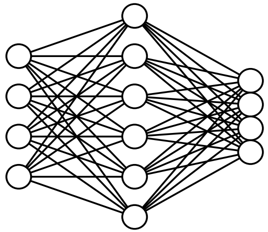
LinnOS with Primo

LinnOS (31 Input Features)

Input



Model



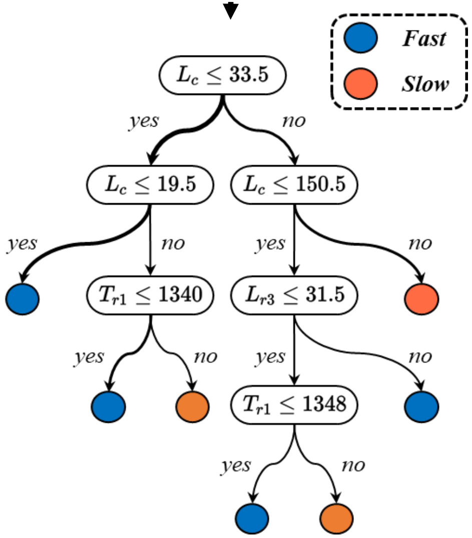
8706 parameters

Output

Fast *Slow*

Primo (3 Input Features)

L_c	Current queue length
L_{r3}	Queue length of the third recent I/O
T_{r1}	Latency of the first recent I/O



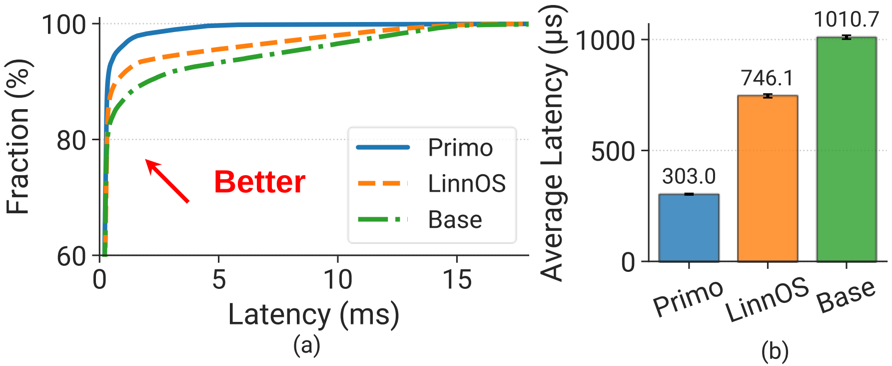
4 layers with 7 leaves

LinnOS: Performance Analysis

- **Overall Performance**

  Average I/O latency:

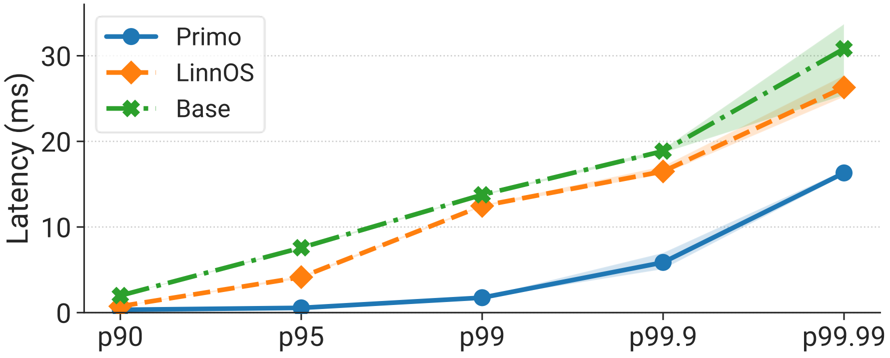
  2.5x reduction compared to LinnOS



- **Tail Performance**

  Tail I/O latency:

  2.2~7.9x reduction compared to LinnOS



LinnOS: Effectiveness Analysis

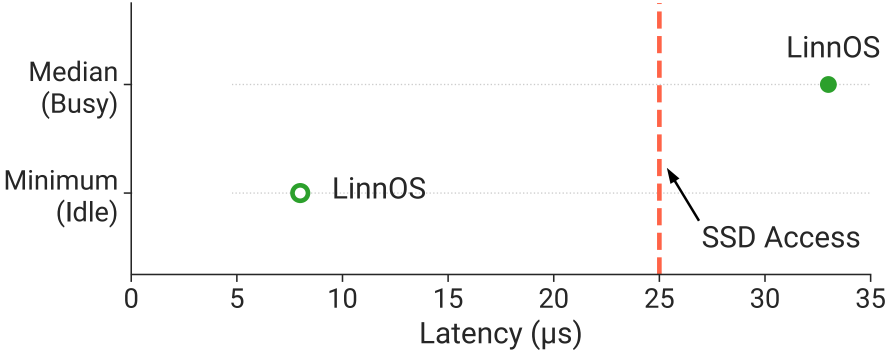
- Inference overhead

  LinnOS: Data Preprocess + DNN Inference

  8 us (idle) 33 us (busy)

  Primo: 4 if-else Condition Tests

  <1 us (idle) 2 us (busy)

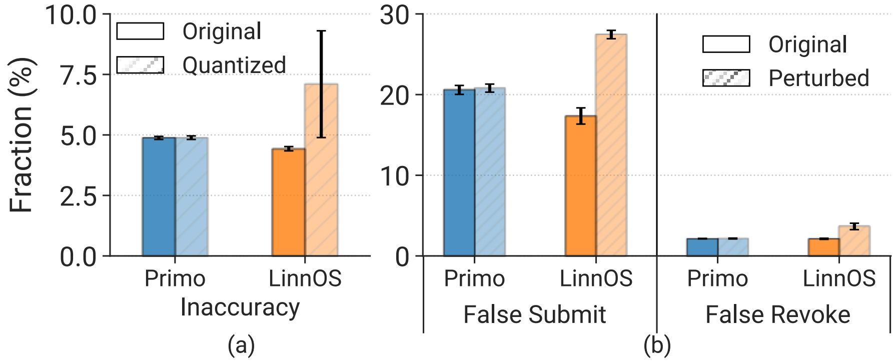


- Quantization

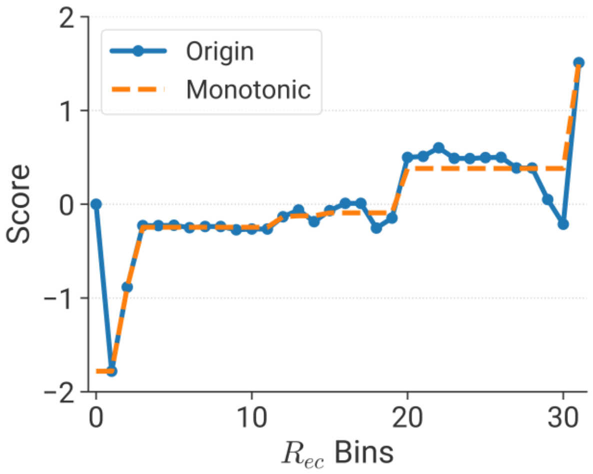
  No degradation and higher accuracy

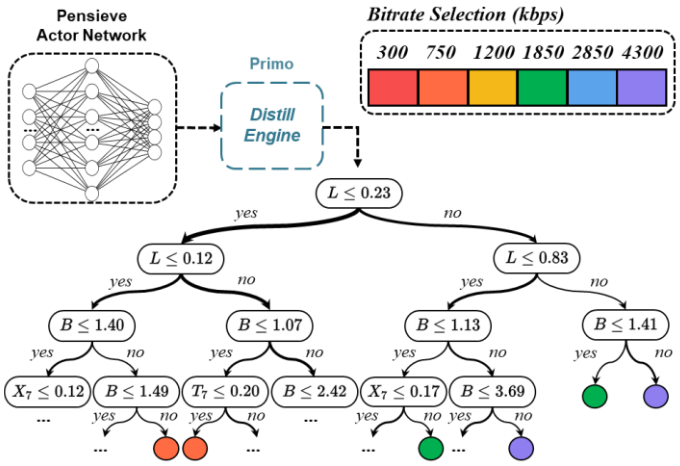
- Robustness

  More stable to the perturbed inputs



More Evaluations





Monotonic Constraint

Clara

Distill Engine

Pensieve

Summary

More Details in Our Paper

PRIMO: Practical Learning-Augmented Systems with Interpretable Models

Qinghao Hu[1,2] Harsha Nori[3] Peng Sun[4] Yonggang Wen[1] Tianwei Zhang[1]

[1]*Nanyang Technological University* [2]*S-Lab, NTU* [3]*Microsoft* [4]*SenseTime Research*

Abstract

While machine learning has demonstrated remarkable performance in various computer systems, some substantial flaws can prohibit its deployment in practice, including opaque decision processes, poor generalization and robustness, as well as exorbitant training and inference overhead. Motivated by these deficiencies, we introduce PRIMO, a unified framework for developers to design practical learning-augmented systems. Specifically, (1) PRIMO provides two interpretable models (PrAM and PrDT), as well as a *Distill Engine*, to support different system scenarios and deployment requirements. (2) It adopts *Bayes Optimization* to automatically identify the optimal model pruning strategy and hyperparameter configuration. (3) It also implements two tools, *Monotonic Constraint* and *Counterfactual Explanation*, to achieve transparent debugging and guided model adjustment. PRIMO can be applied to different types of learning-augmented systems. Evaluations on three state-of-the-art systems show that PRIMO can provide clear model interpretations, better system performance, and lower deployment costs.

1 Introduction

Over the years, machine learning (ML) has been widely adopted to optimize systems across many fields, e.g., storage [29,82,85], network [66,77,95], security [24,28,74], compiler optimization [8,93,94] and cluster scheduling [65,89,92]. These learning-augmented systems demonstrate marvelous performance compared with conventional heuristic or mathematical optimized systems.

However, most of these applied models are very complex and treated as black-boxes to developers, which brings significant gaps in deploying them in practice. **First, building a production-level learning-augmented system can incur huge costs.** From the experience at Microsoft [42], the model training process could take days to weeks with massive data. Some systems require frequent model updates to adapt to dynamic environment changes, whose cost often exceeds en- systems which have high real-time requirements [43,81,82], which can significantly restrict parallel capabilities and affect scalability in practice.

Second, the prediction process of these black-box models are unintelligible to humans. Developers lack understanding and trust of the model's behavior [19,53,91], which makes it difficult for them to perform model adjustments and ad hoc debugging in practical scenarios. Some efforts have been made to improve system transparency through interpreting black-box models [26,27,55]. They typically build *surrogate* models to obtain explanations for individual predictions, thus validating model behaviors and diagnosing system mistakes. However, they cannot provide an interpretation fidelity guarantee, and therefore the corresponding explanations are unreliable and potentially misleading [58,70]. In addition, they cannot address the aforementioned system cost issue.

In this paper, we aim to resolve the above challenges and facilitate *transparent*, *accurate* and *lightweight* system deployment in practice. We introduce PRIMO (Prior-based interpretable model optimization), the *first* unified framework that assists developers to design and optimize learning-augmented systems with interpretable models. The design of PRIMO is based on two key insights. First, *simple interpretable models have the capability of handling complex system problems.* Interpretable models do not sacrifice prediction accuracy [35,62,72], and simple model structures with low resource overhead are very suitable for real-time systems. Their effectiveness is often underestimated [70]. Second, *prior experience and domain knowledge can be leveraged by developers to further optimize the interpretable models* [20,76], which is hard to achieve for black-box models.

PRIMO makes several innovations to enhance learning-augmented systems. First, to provide comprehensive support for different systems, PRIMO introduces two interpretable model algorithms: PrAM is designed for better prediction accuracy and PrDT applies to systems with strict latency or computation constraints. PRIMO can help developers select a suitable

Summary

- Non-trivial to deploy learning-augmented systems in practice

  Training cost **Inference overhead** **Data insufficiency**

  Opaque decision **Hard to adjust …**

- Simple interpretable models are excellent choices

  We demonstrate they can outperform original black-box models in LinnOS, Clara and Pensieve

- Operators need automatic and guided model optimization

Our Code is Open Source:

https://github.com/S-Lab-System-Group/Primo