Primo: Practical Learning-Augmented Systems with Interpretable Models

Qinghao Hu¹, Harsha Nori², Peng Sun³, Yonggang Wen¹, Tianwei Zhang¹
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

Prototype → More challenges in practice → System

Research Testbed → Production Environment
Challenges in Practical Deployment

Prototype

Research Testbed

System

Production Environment
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
Strict Inference Overhead Requirement

Latency Constraint

- **AI Apps**: ~10ms
- **ML-Sys**: ~10μs

Resource Constraint

- **CPU**
- **Memory**
- **Storage**

- **Limited Scalability**
  - Testbed-scale: ✔️
  - Production-scale: ✗

- **Side-effect to Production Workloads**
  - ML models occupy too much resources

**Diagram**

1. **Cost**
   - Research Testbed
   - Prototype
   - Overhead

2. **System**
   - Production Environment
Insufficient Data

ML Model is Data-Driven

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

More complex model needs more data!
Opaque Decision Process

Many Learning-Augmented Systems Rely on Black-Box Models

- **Interpretability is Important But Often Ignored**
  Operators need sufficient confidences to deploy learning-augmented systems.

- **Data**
  - System States
  - Workload Features
  - User Configurations

- **Black-box Models**
  - DNNs

- **Decision**
  - Scale Up?
  - Schedule?
  - Terminate?

- **How the model make decision?**
- **Should I trust the prediction?**

**Prototype**
- **Cost**
- **Overhead**
- **Opacity**

**System**
- **Research Testbed**
- **Production Environment**

1. Cost
2. Overhead
3. Data
4. Opacity
Hard to Adjust

System Adjustment is Needed

Too Complex for System Operators

Different: System Scale Machine Type

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

<table>
<thead>
<tr>
<th></th>
<th>Interpretation</th>
<th>Training Cost</th>
<th>Inference Overhead</th>
<th>Insufficient Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Individual</td>
<td>Entire</td>
<td>Fidelity</td>
<td></td>
</tr>
<tr>
<td>Interpreting Black-box</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th></th>
<th>Inherently Intelligible</th>
<th>Simple &amp; Lightweight</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Interpretation</td>
<td>Training Cost</td>
</tr>
<tr>
<td></td>
<td>Individual</td>
<td>Entire</td>
</tr>
<tr>
<td>Interpreting Black-box</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Interpretable Model</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Why Interpretable Models Work

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

Key Observations

1. Input feature

<table>
<thead>
<tr>
<th>AI Apps</th>
<th>ML-Sys: System States</th>
<th>Workload Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Pixels</td>
<td>Meaningful and Lower Dimensional</td>
<td></td>
</tr>
<tr>
<td>Word Embeddings</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Model Scale

<table>
<thead>
<tr>
<th>AI Apps</th>
<th>ML-Sys: RL-Sys</th>
<th>&lt;10K neurons¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18</td>
<td>Smaller Scale and Latency Sensitive</td>
<td></td>
</tr>
<tr>
<td>BERT-Base</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 clearly 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
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)**

- \( L_c \) with 3 neurons
- \( L_{r1} \ldots L_{r4} \) with 3×4 neurons
- \( T_{r1} \ldots T_{r4} \) with 4×4 neurons

**Output**

- Fast
- Slow

**Model**

8706 parameters

**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

- \( L_c \leq 33.5 \)
- \( L_c \leq 19.5 \)
- \( L_c \leq 150.5 \)
- \( T_{r1} \leq 1340 \)
- \( L_{r3} \leq 31.5 \)
- \( T_{r1} \leq 1348 \)
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
PRIMO: Practical Learning-Augmented Systems with Interpretable Models

Qinghao Hu\textsuperscript{1,2} Harsha Nori\textsuperscript{3} Peng Sun\textsuperscript{4} Yonggang Wen\textsuperscript{1} Tianwei Zhang\textsuperscript{1}

\textsuperscript{1}Nanyang Technological University \textsuperscript{2}S-Lab, NTU \textsuperscript{3}Microsoft \textsuperscript{4}SenseTime Research

Abstract

While machine learning has demonstrated remarkable performance in various computer systems, some substantial flaws can prohibit their deployment in practice, including opaque decision processes, poor generalization and robustness, as well as extravagant 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 (PAPR and P2T), as well as a Distill Engine, to support different system scenarios and deployment requirements. (2) It adopts Bases Optimization to automatically identify the optimal model pruning strategy and hyperparameter configurations. (3) It also implements two tools, Monotonic Constraints and Counterfactual Explanations, 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 [20,82,84], network [46,77,97], security [24,38,74], compiler optimization [8,39,94] 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 incurs 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 the 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 [19,33,91], 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 surrogates 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 (P2T-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 [59]. 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: PAPR is designed for better prediction accuracy and P2T 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