WiP: Searching Optimal Compiler Optimization Passes Sequence for Reducing Runtime Memory Profile using Ensemble Reinforcement Learning

Juneseo Chang[1] and Daejin Park[3]

[1] Department of Computer Science and Engineering, Seoul National University, Korea

[2] School of Electronics Engineering, Kyungpook National University, Korea

AI-**E**mbedded
Software **o**n **C**hip Lab.

Introduction

- Compiler-level code optimization is important but difficult
  - Optimizations are applied as a sequence of transformation passes

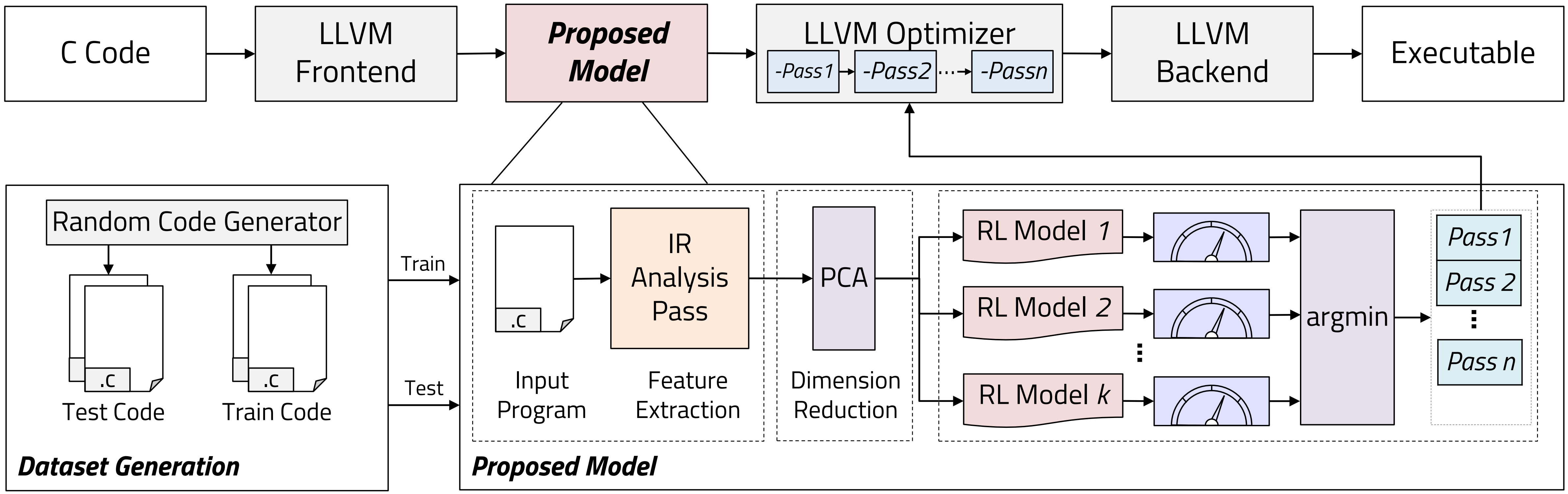
`-fauto-inc-dec` → `-fbranch-count-reg` → `-fcombine-stack-adjustments` → … → `-ftree-ter` → `-funit-at-a-time`

GCC –o3

  - However, a single *passes sequence* are selected that are efficient on specific benchmark
  - Performance gap exists between the ideal and actual compiler-optimized programs
- Among various optimization purposes, we focus on memory profile reduction
  - To the best of our knowledge, there is no compiler-level optimization for memory profile

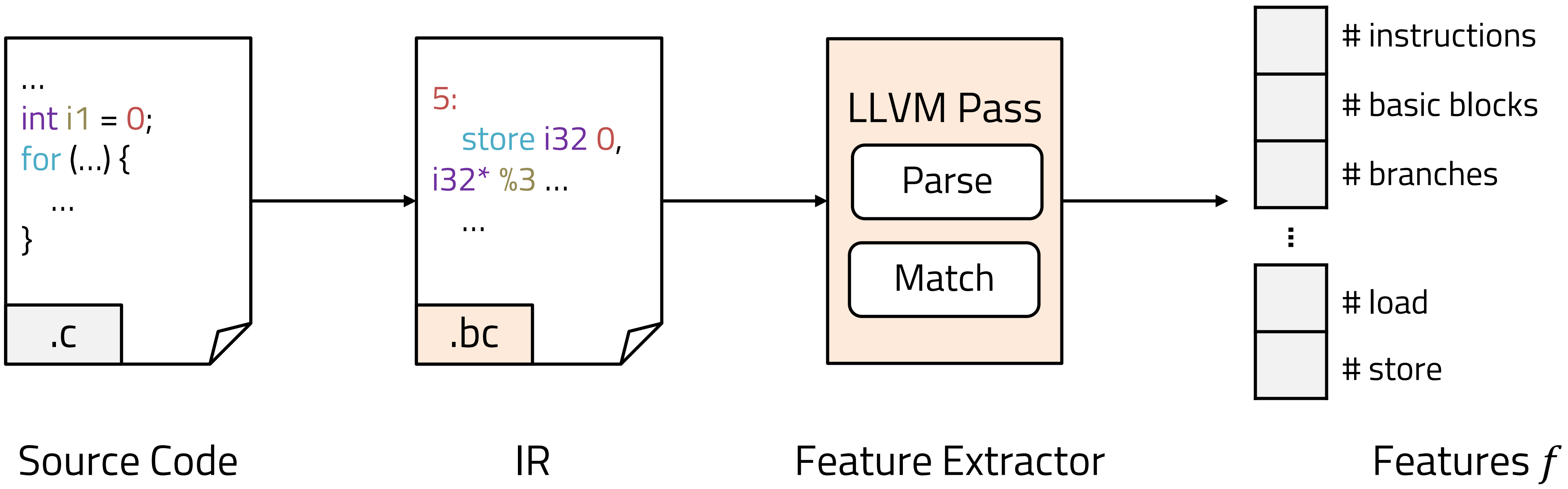
M. Hall, D. Padua, and K. Pingali, "Compiler research: The next 50 years," Commun. ACM, vol. 52, no. 2, p. 60–67, feb 2009.
A. H. Ashouri, W. Killian, and et al., "A survey on compiler autotuning using machine learning," ACM Comput. Surv., vol. 51, no. 5, sep 2018

Overview



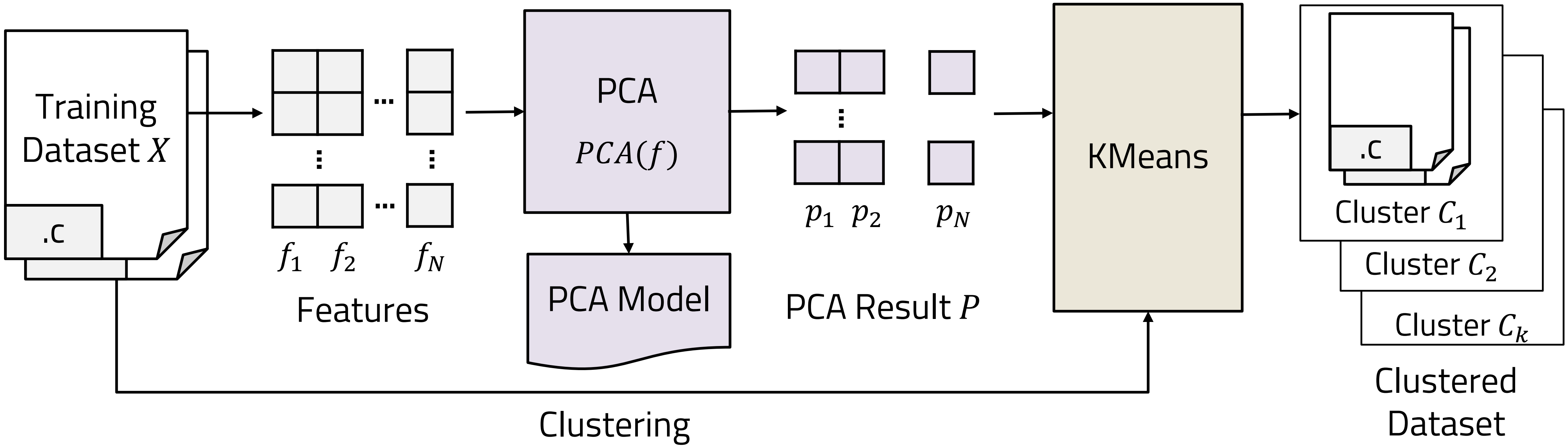
C Code
LLVM Frontend
Proposed Model
LLVM Optimizer
-Pass1
-Pass2
-Passn
LLVM Backend
Executable
Random Code Generator
.c
.c
Test Code
Train Code
Dataset Generation
Train
Test
.c
Input Program
IR Analysis Pass
Feature Extraction
PCA
Dimension Reduction
RL Model 1
RL Model 2
RL Model k
argmin
Pass1
Pass 2
Pass n
Proposed Model

Feature Extraction and Dataset



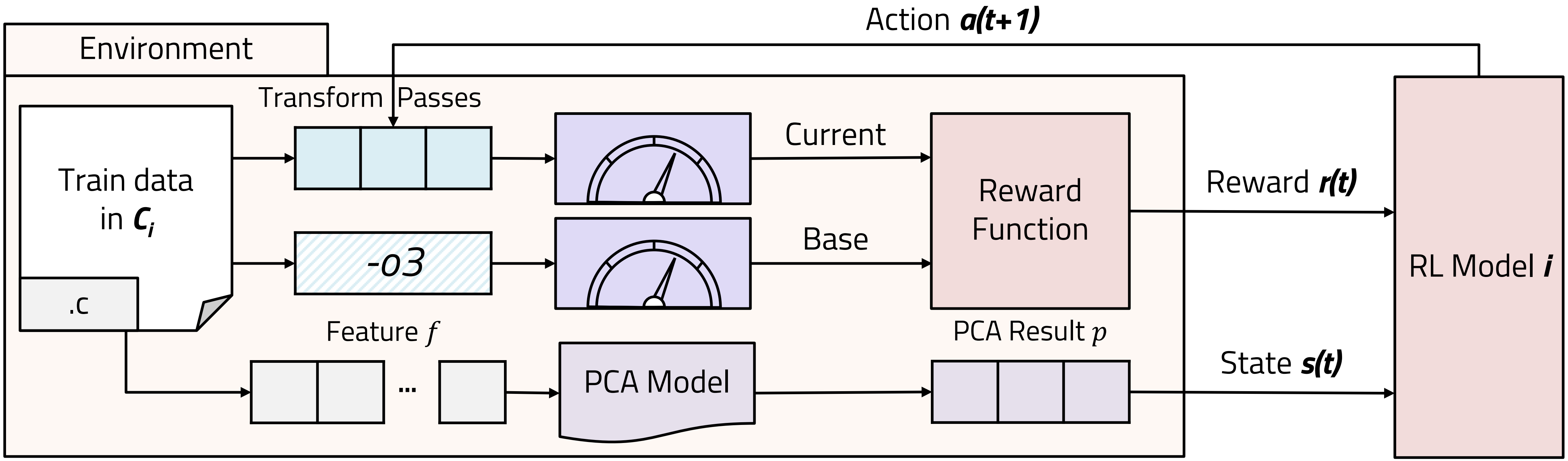
- **Feature Extraction.** Extract 50 static feature at compile time from IR with LLVM IR analysis pass
- **Dataset.** Generate valid random C codes with code sizes over 300kb using Csmith
  - Existing microbenchmarks were too simple to reduce memory profile
- **Transform Pass Candidates.** Selected 32 LLVM transform passes that reduce memory profile

Dimension Reduction and Clustering



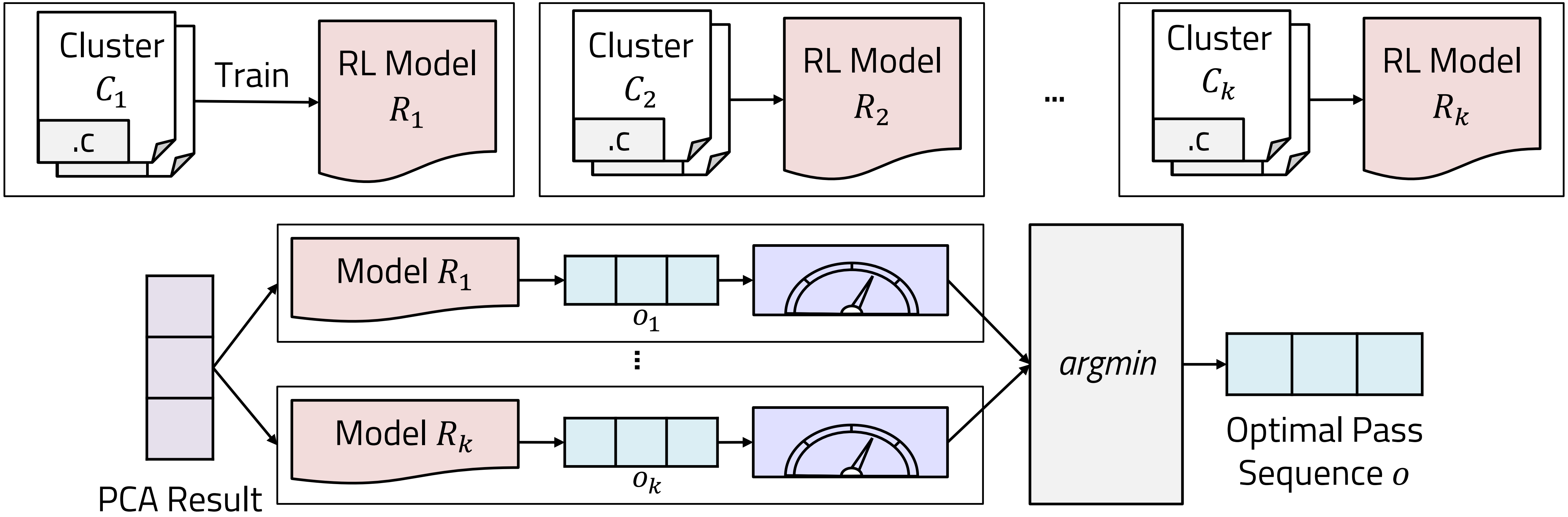
- **PCA.** Applied PCA to reduce dimension to 15 components, accounting for 99% variances.
- **Clustering.** Clustered PCA results into 4 subsets by Kmeans
  - Alleviate the unfitting and reduce the parameter number for each model

Ensemble RL Model



- RL model learns the optimal transform pass that maximizes memory reduction
  - **State.** PCA result
  - **Reward.** $r = \frac{m_s - m_0}{m_0}$
  - **Action.** each entry representing a transform pass in order
  - **Learning Algorithm.** PPO, A3C, PG

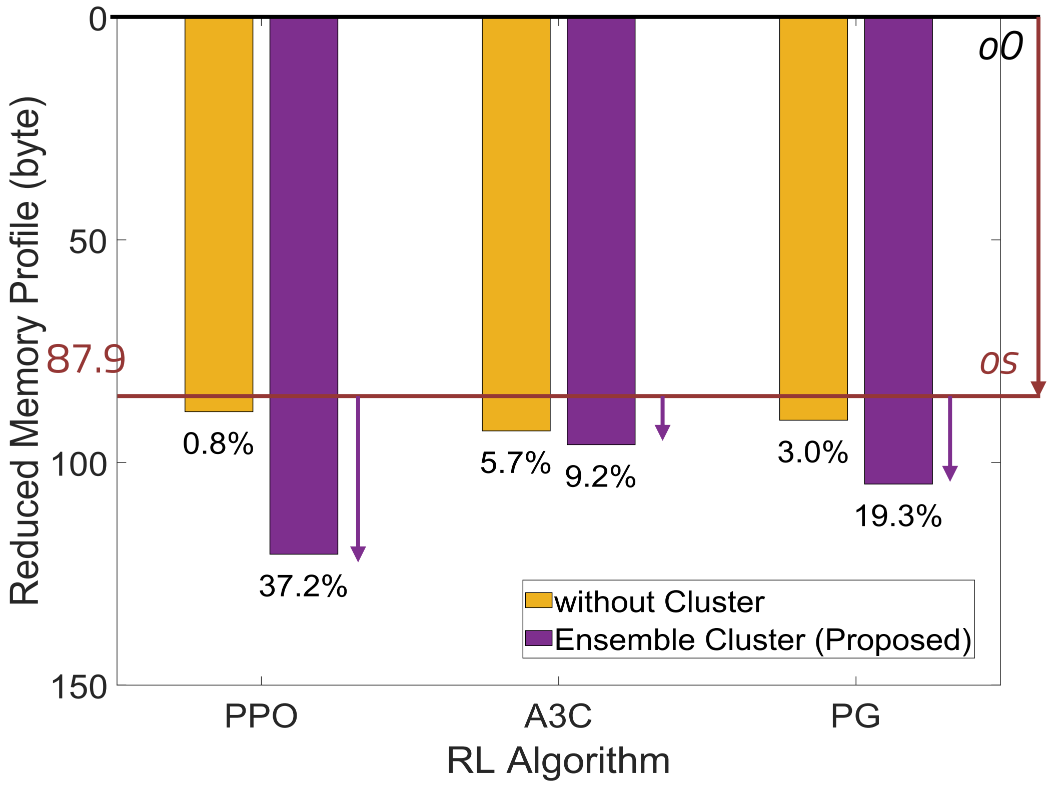
Ensemble RL Model



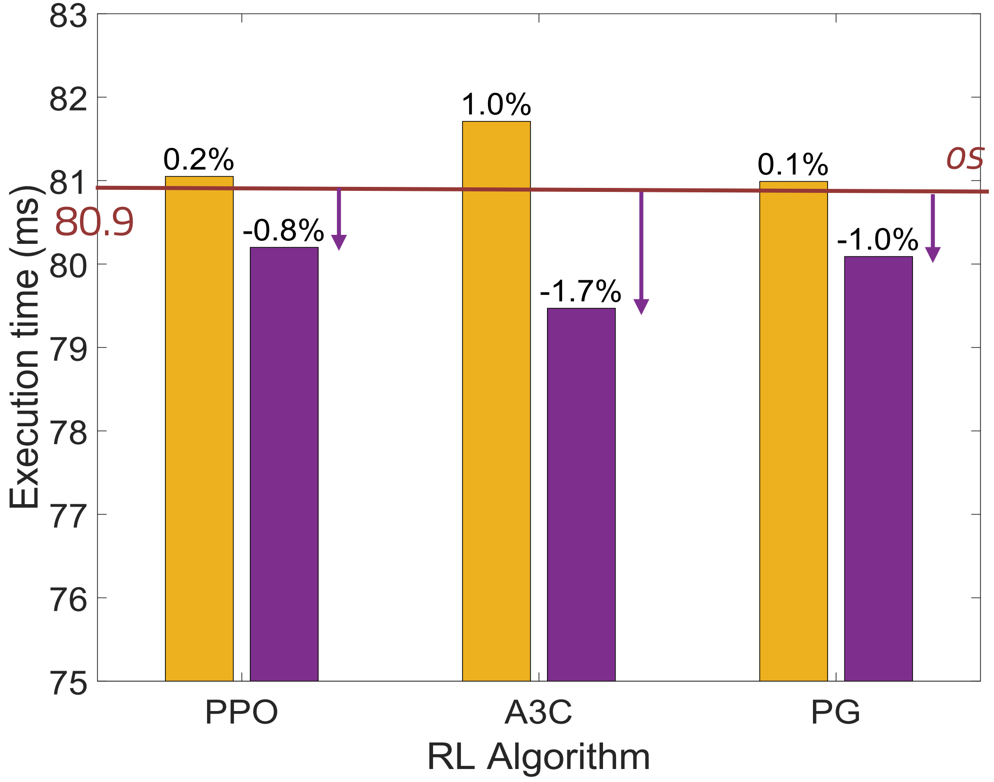
- Capturing the behavior of various applications with a one-size-fits-all model is inefficient
  - We propose ensemble learning by training each model on a different clustered dataset
- Among the outputs of each RL model, output best minimizing memory profile was chosen
  - Execution time is used as tie-breaker

AI-**E**mbedded **S**oftware **o**n **C**hip Lab.

Experimental Result



Reduced peak memory profile



Execution time

- Implemented ensemble of clustered datasets, non-ensemble models for each learning algorithm
- Up to 37% memory profile reduction, while also reducing the execution time

Conclusion and Outlook

- **Summary.** Proposed an ensemble RL model that exploits clustered datasets to find the transform passes sequence that reduced 37% more memory profile compared to the LLVM -os option.
- **Contribution.** Expect embedded system developers to easily improve the program's memory profile without "human-in-the-loop" processes.
- **Plan.** In-depth analyze the RL model behavior, Evaluate on real-world, feasible application

Q & A

Thank you for your attention

Juneseo Chang: jschang0215@snu.ac.kr

Available on GitHub