MAPP: Predictive UI View Pre-caching for Improving the Responsiveness of Mobile Apps

Run Wang, Zach Herman, Marco Brocanelli, Xiaorui Wang

Dept. of Electrical and Computer Engineering

The Ohio State University

Power-Aware Computer Systems (PACS) Lab

Introduction

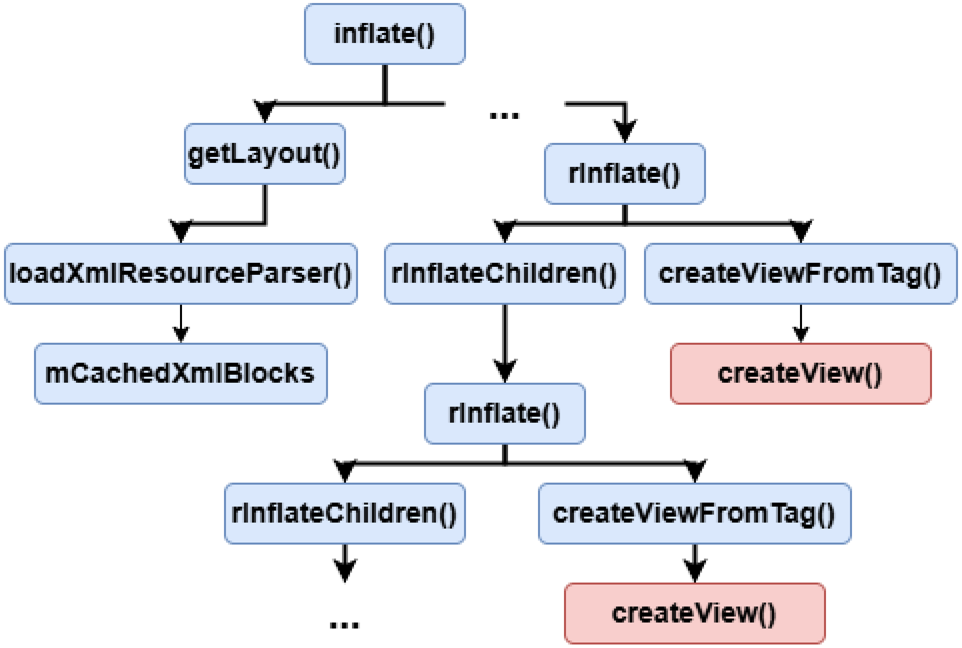
- 8.93 million mobile apps (2023); Americans spend ~5 hours/day on them.
- Delays >2–3 seconds lead to app abandonment.
- Hardware upgrades help, but software-induced delays persist.

Introduction

- Two major causes of soft hangs (EuroSys '18):
  - Hang bugs (well-studied)
  - Prolonged UI-APIs (underexplored)

Introduction

- Creating UI view hierarchy is the main bottleneck (~60% of UI-API time)

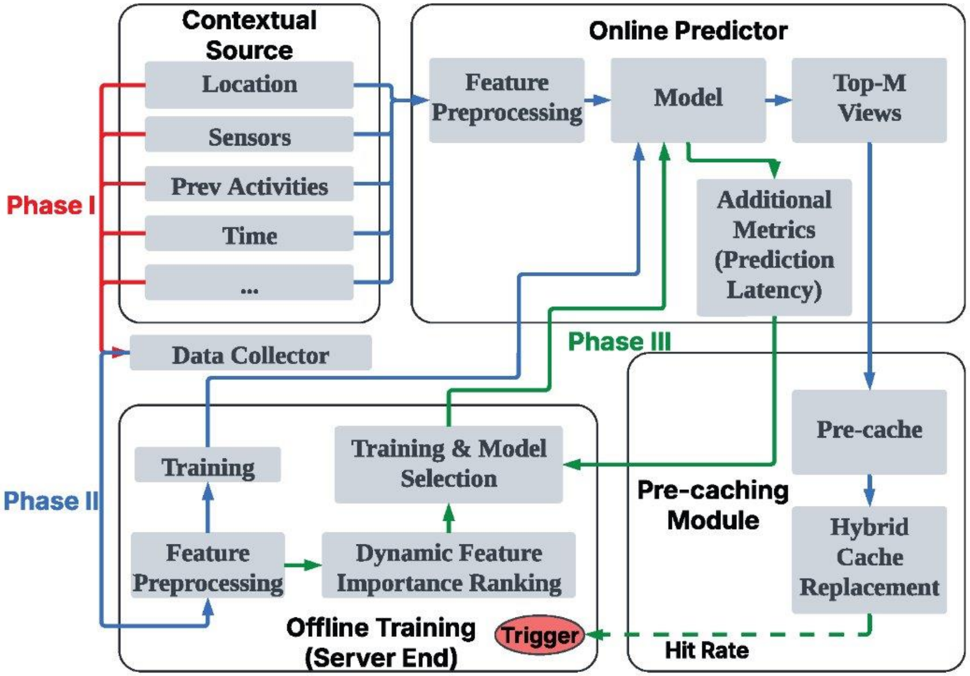


Motivations

- Regular caching helps only for repeat views; first visit remains slow
- Pre-caching every possible view:
  - Consumes time, energy, and memory
- Need selective, predictive caching to balance trade-offs

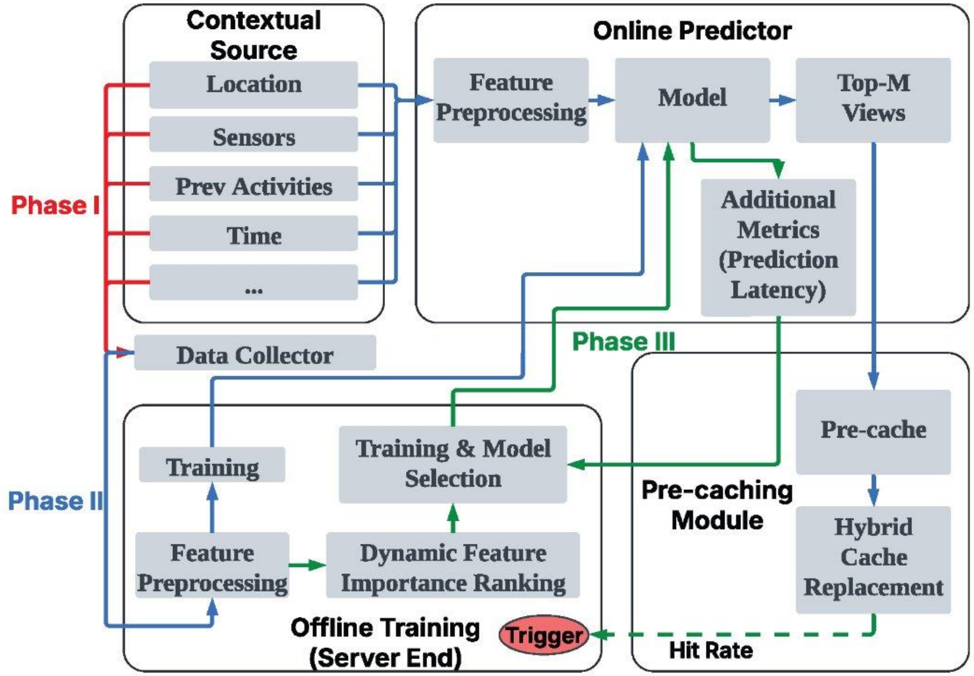
Solution: MAPP Framework

- Predicts likely next UI view per user and per app
- Pre-caches top views to accelerate UI rendering



Solution: MAPP Framework

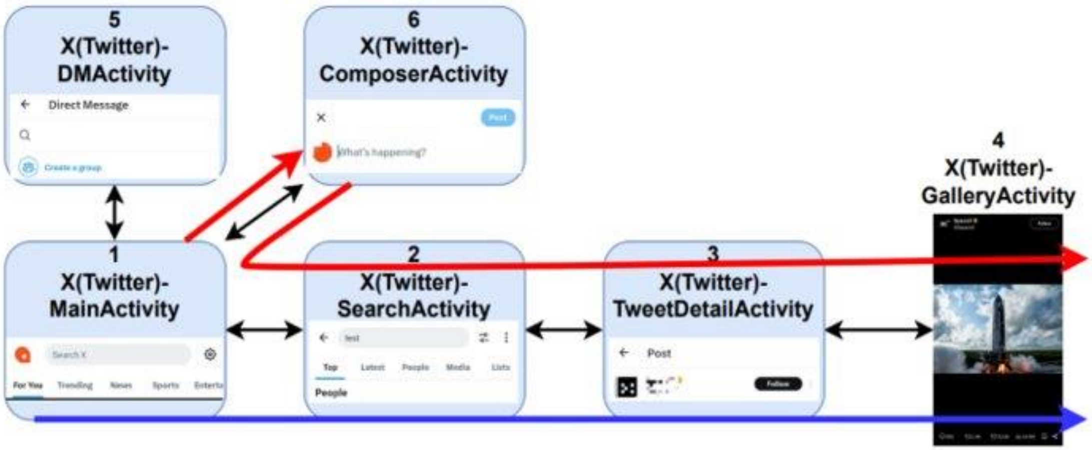
- **Phase I**: Collect user traces (location, time, view sequence)
- **Phase II**: Train prediction model → start pre-caching
- **Phase III**: Feature importance ranking + model optimization



THE OHIO STATE UNIVERSITY

View Prediction Input Features

- Sequential features: History of previously visited UI views.
- Contextual features: Location, time of day, battery level, signal strength, sensor data.

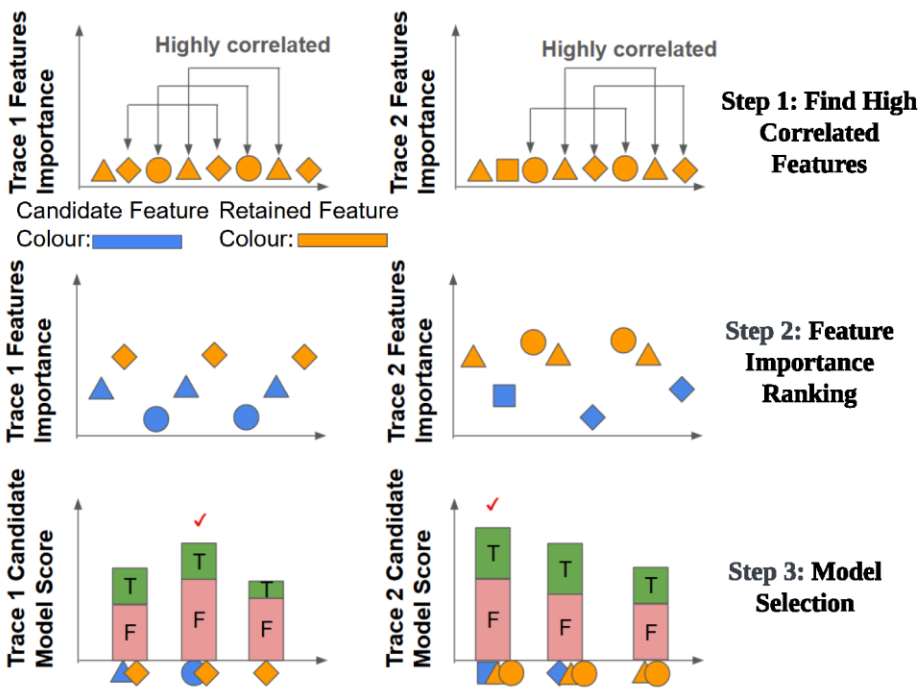


Model Training Design

- Uses (Gated Recurrent Unit) GRU neural networks for efficient sequence modeling.
- Dense layers process contextual data separately.
- Incorporates dropout layers to prevent overfitting, class weighting for handling imbalanced datasets.

Feature and Model Selection

- Modified permutation importance ranks features effectively.
- Groups correlated features.
- Excludes less impactful groups while preserving accuracy.



Feature and Model Selection

- Objective: select the model with highe
  - Score(M) = F_measure(M) - λ * Time(M)
  - Time(M) is estimated end-to-end latency associated with Model
- Balances prediction accuracy against latency.
- Selects models based on runtime constraints and accuracy requirements.
- Optimizes overall responsiveness without exceeding mobile latency budgets.

View Cache Design

- Pre-caches Top-M predicted UI views (default M=3).
- Utilizes a hybrid replacement policy: predictive pre-caching and fallback LRU for unexpected patterns.
- Manages cache efficiently to minimize overhead and maximize hit rates.

Android Implementation

- Replace UI-APIs in **onCreate()** with cache-first logic

```
function PRE-CACHE
    inflater ← getLayoutInflater()
    view ← inflater.inflate(R.layout.predicted_layout, null)
    ViewCache.add(view)
end function
```

```
function ONCREATE
    view ← ViewCache.get(localClassName)
    if view ≠ null then
        setContentView(view)
    else
        inflater ← getLayoutInflater()
        view ← inflater.inflate(R.layout.layout_dummy, null)
        setContentView(view)
    end if
end function
```

Experimental Setup

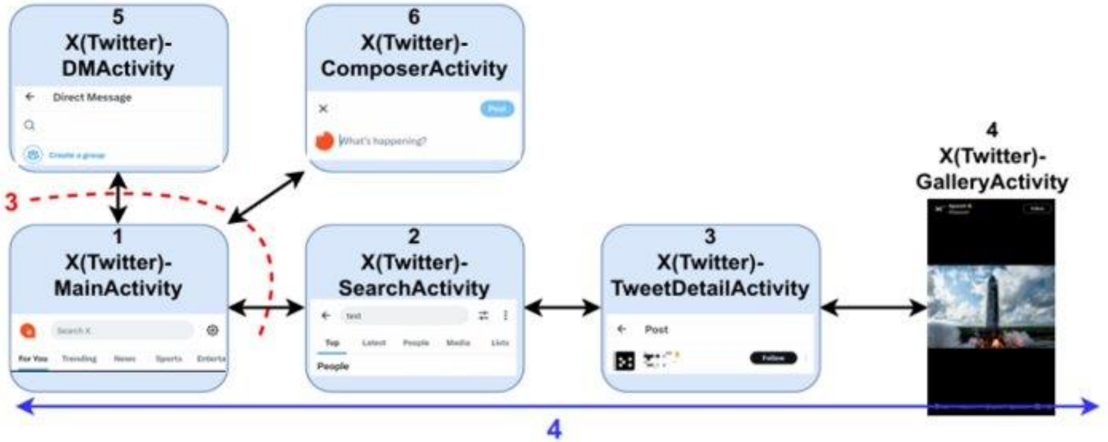
- Evaluated using Google Pixel 8 and Samsung Galaxy A11.
- Data: 61 traces from 18 users over 30 days, capturing diverse real-world usage patterns.
- Traces contain detailed user actions, contextual data (location, sensors, etc.), and app usage sequences.

EXAMPLES OF APP USAGE DATA

Activity Name	Time	Location	...	Battery Level	Signal Strength	Duration (ms)
Main	161553	39, -82	...	88	1	1133
Search	161600	39, -82	...	88	1	7161
Profile	161618	39, -82	...	87	1	17821
Gallery	161622	39, -82	...	87	1	3814

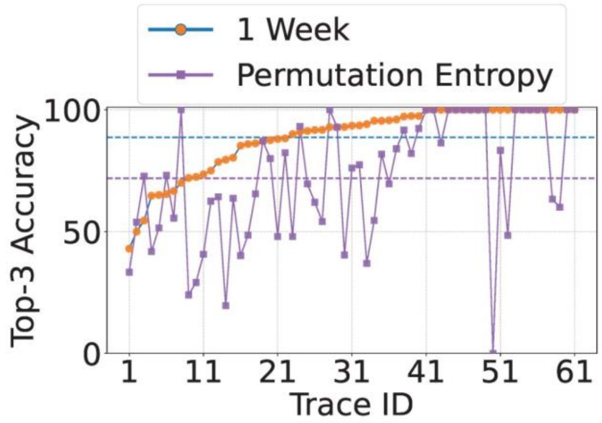
Dataset Characteristics

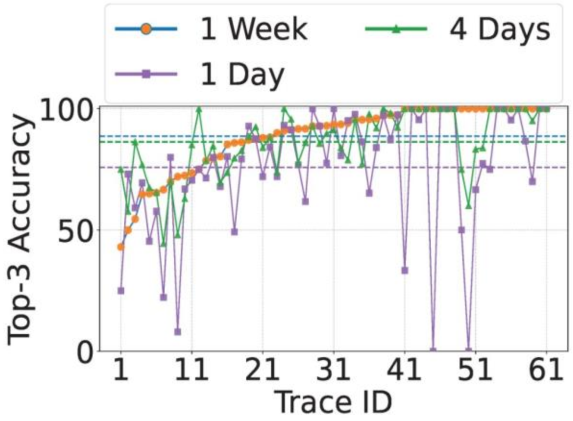
- Avg. max view depth: ≥4 → long sequences
- Avg. view branching factor: 6+ → naïve caching inefficient
- Complex pattern shows that naïve caching is inadequate.



Evaluation

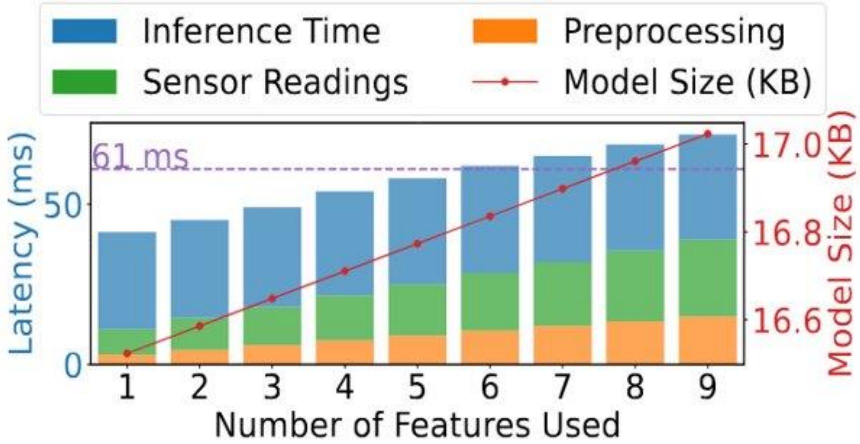
- Evaluates various data-collection periods (1-day vs. 1-week) and entropy-based approaches.
- One-week data collection provides highest overall prediction accuracy.





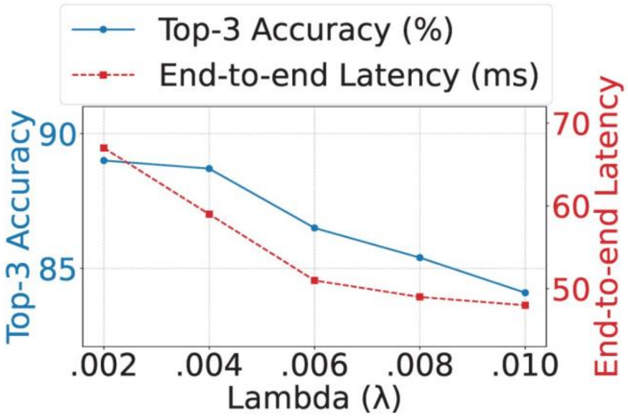
Evaluation

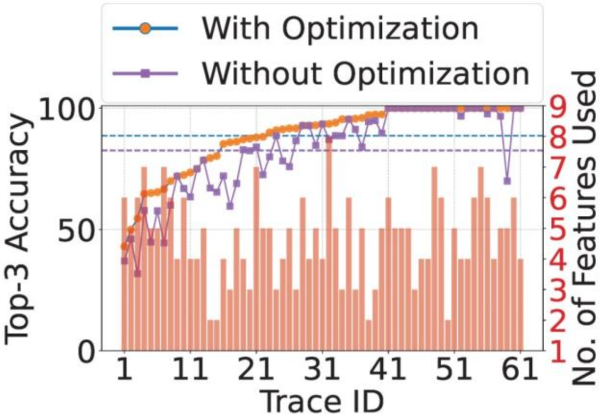
- End-to-End Latency Breakdown
- Fewer features → less latency & smaller model



Evaluation

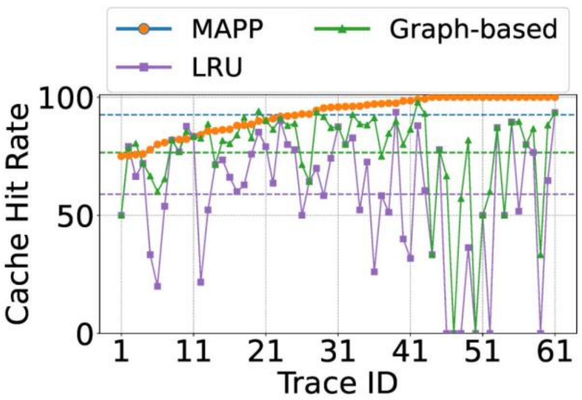
- Optimization improves top-3 prediction accuracy from 82.6% to 88.7%.
- Effectively balances accuracy with runtime constraints through parameter tuning (λ).

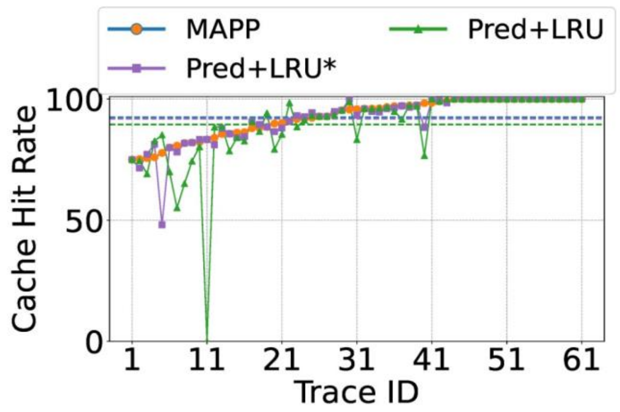




Evaluation

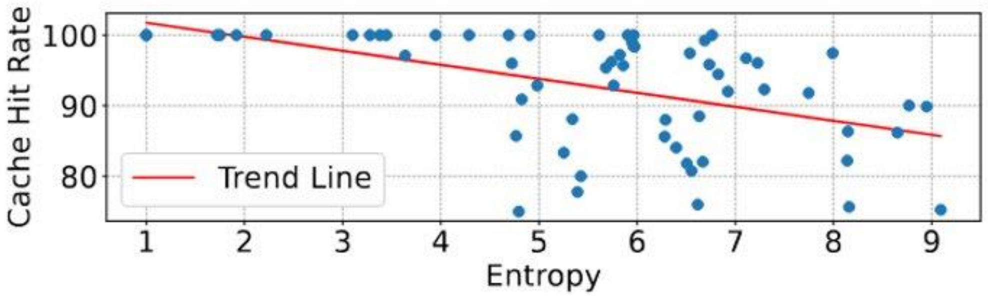
- MAPP hit rate: 92.55%
- vs. LRU: 58.95%, Graph-based: 76.55%





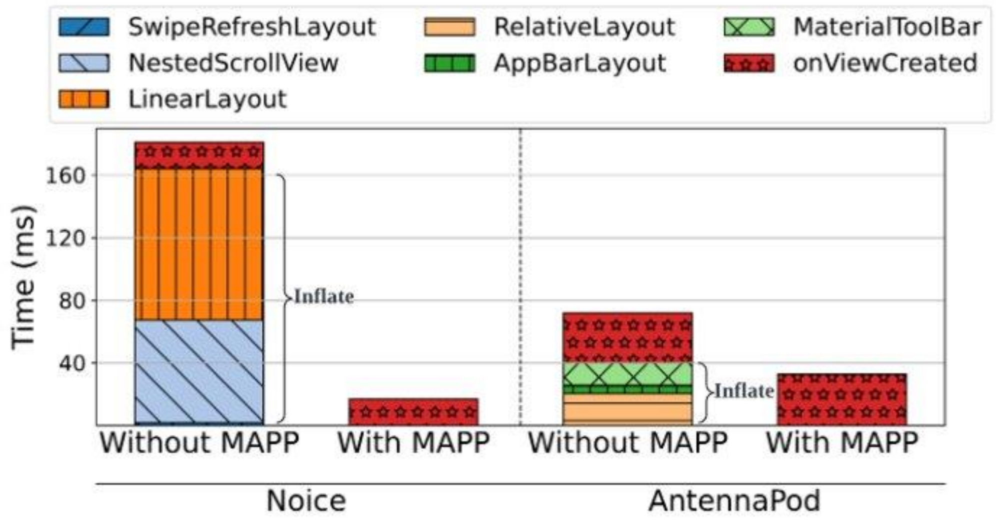
Evaluation

- Higher permutation entropy in user interactions reduces cache hit rate.
- Highlights challenges of predicting complex user interactions.
- Future research should focus on advanced prediction refinement for high-entropy cases.



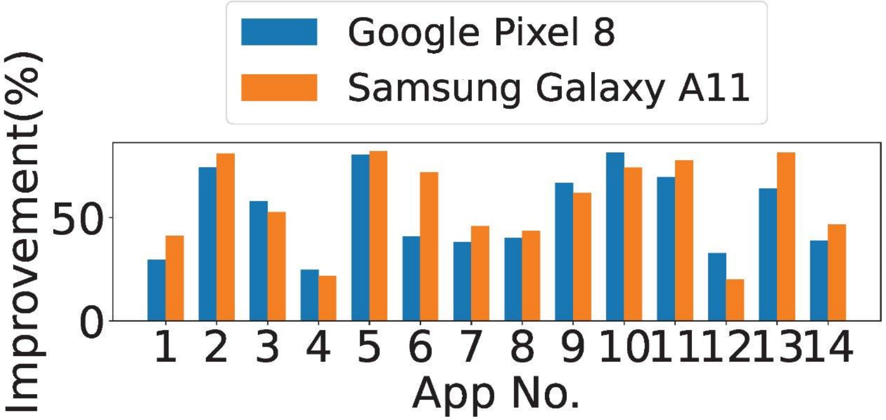
Evaluation

- In-depth breakdown demonstrates substantial latency contribution from executing UI-APIs.
- Pre-caching directly targets and significantly mitigates this primary bottleneck.



Evaluation

- Tested on multiple real-world apps, latency reduction averages ~60%.
- Substantial gains on lower-end devices (Galaxy A11) emphasize broader applicability.
- Latency reductions especially impactful in apps with complex UI hierarchies.



Evaluation

- Overall responsiveness improvement between 53-57% across devices.
- Modest overhead: Power usage increases ~3.27%, memory increases ~2.64%.
- Demonstrates MAPP's practicality in real-world mobile scenarios.

Conclusion

- MAPP significantly improves overall UI responsiveness
- Smart pre-caching + model optimization = practical gains
- Can generalize across apps and devices
- Future Work: Integrate with content prefetching, extend to Jetpack Compose