Leveraging gem5 and Machine Learning for End-to-End Detection of Cache-based Side-Channel Attack Patterns

PROOFs 2025

Muhammad Awais

Télécom Paris IP Paris

Plan

1. Introduction
2. Gem5 for Attack Patterns
3. Attack Pattern Clustering Using ML

Introduction

MUHAMMAD AWAIS
1st Year Phd Student at Telecom Paris

LIRMM
Work with the gem5 simulations

FAST Islamabad, Pakistan
Masters in Computer Science

Gem5

Gem5 tool for micro architecture security research.

- Gem5 is a customizable, open-source computer system simulator widely used for architectural research, enabling the modeling and simulation of diverse computer systems.

Simulation Modes.

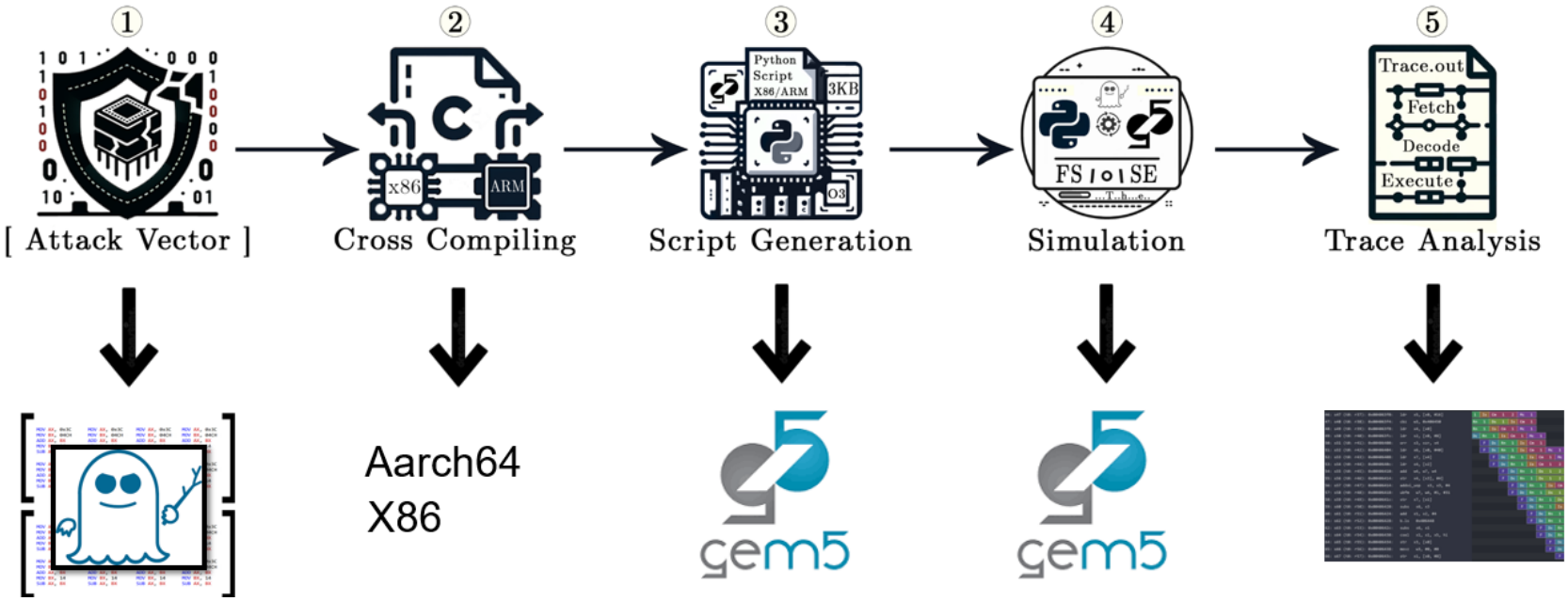
- **Full-System (FS) Mode**, which simulates an entire system including hardware and an OS, useful for studying OS-hardware interactions.
- **System Call Emulation (SE) Mode**, which emulates system calls and simulates only user-level execution for faster application-level performance studies.

CPU Models Support in gem5 (Hardware & Features)

CPU Models in gem5

CPU Model	Type	Description
MinorCPU	In-order	Simple in-order model, used for embedded-like simulations.
TimingSimpleCPU	In-order	Basic timing model with simple execution.
DerivO3CPU	Out-of-order	More advanced out-of-order core model for research.
AtomicSimpleCPU	Atomic (Fast)	No detailed execution, used for quick functional validation.

Pattern Generation Methodology



Attack Pattern Clustering Using ML

Attack Patterns

Speculative, Prime+Probe, Flush+Reload, Normal execution

```
TARGET_ADDR ((char*)0x7ffff7dd18e0)
unsigned long time1, time2;
volatile char *addr = TARGET_ADDR;
_mm_clflush(addr); // Flush from cache
time1 = __rdtscp(&junk);
(void)*addr;          // Reload
time2 = __rdtscp(&junk);
if (time2 - time1 < 80) {
    printf("Victim likely accessed the data!\n");
}
```

Sample Flush Reload attack Pattern

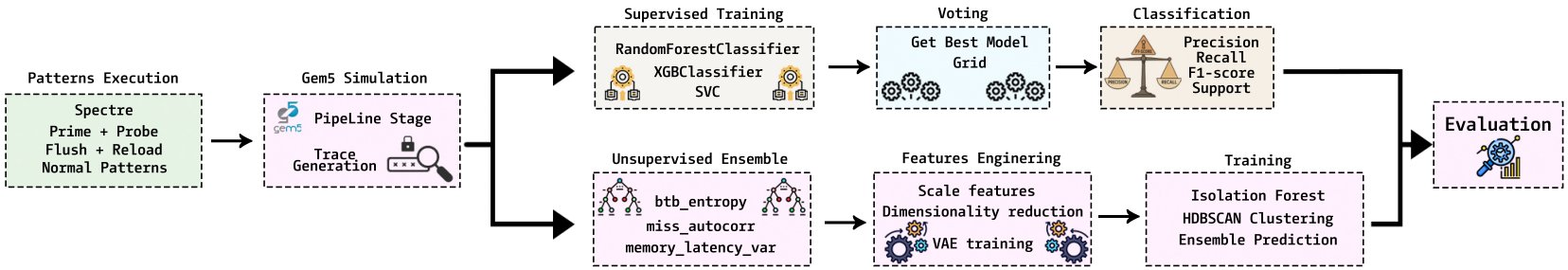
Normal Execution

Listing 1.2. Normal Memory Access Code

```
int arr[1000];
for (int i = 0; i < 1000; i++) {
    arr[i] = i * 2; // Simple arithmetic operation
}
printf("Computation complete.\n");
```

Normal execution Pattern

Methodology

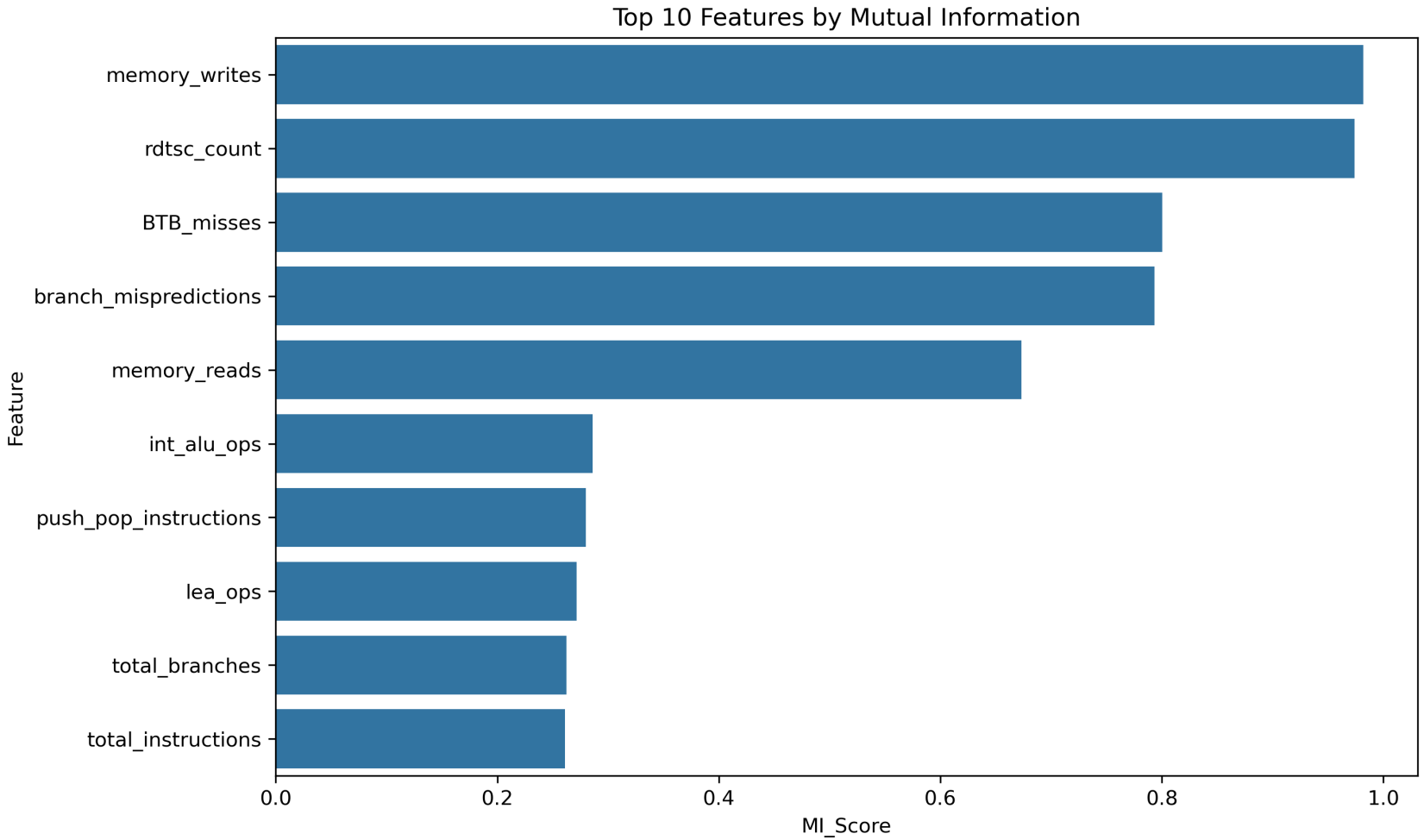


Multi-stage attack classification framework architecture showing the integrated simulation and Clustering pipeline

Features Selection for ML Models

Feature Name	Data Type	Description
total_instructions	integer	Retired instructions count
total_branches	integer	Branch instructions executed
branch_mispredictions	integer	Incorrect branch predictions
memory_reads	integer	Main memory read ops
memory_writes	integer	Main memory write ops
int_alu_ops	integer	Integer ALU ops executed
call_instructions	integer	Function call instructions
push_pop_instructions	integer	Stack ops (PUSH/POP)
rdtsc_count	integer	RDTSC executions (timing)
test_ops	integer	TEST instruction executions
lea_ops	integer	Load Effective Address ops
BTB_misses	integer	BTB miss events
BTB_hits	integer	BTB hit events
BTB_miss_addresses	hex list	BTB miss memory addresses
BTB_hit_addresses	hex list	BTB hit memory addresses
attack_label	categorical	Attack classification label

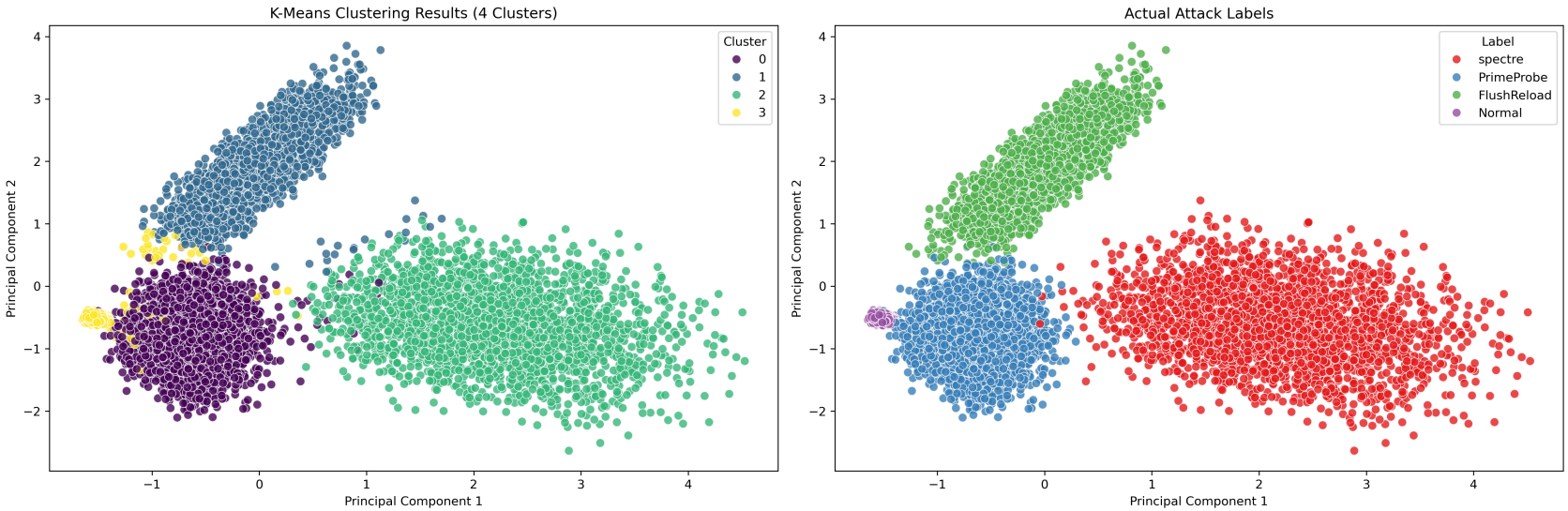
Feature Importance



ML Methodologies Used

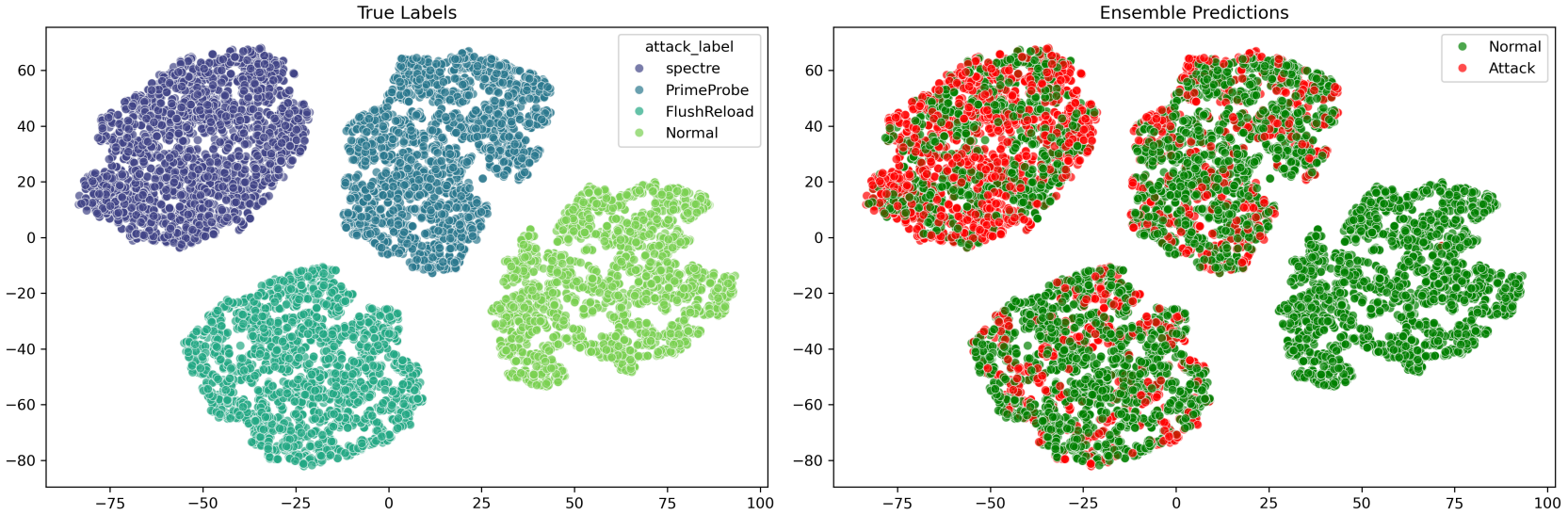
Model Name	Methodology
Isolation Forest	Unsupervised
VAE	Unsupervised
HDBSCAN	Unsupervised
Ensemble	Unsupervised
SVM Baseline	Supervised
XGBoost	Supervised
RandomForestClassifier	Supervised

K Mean Clustering



KMeans Clustering Results and Actual Attack Labels

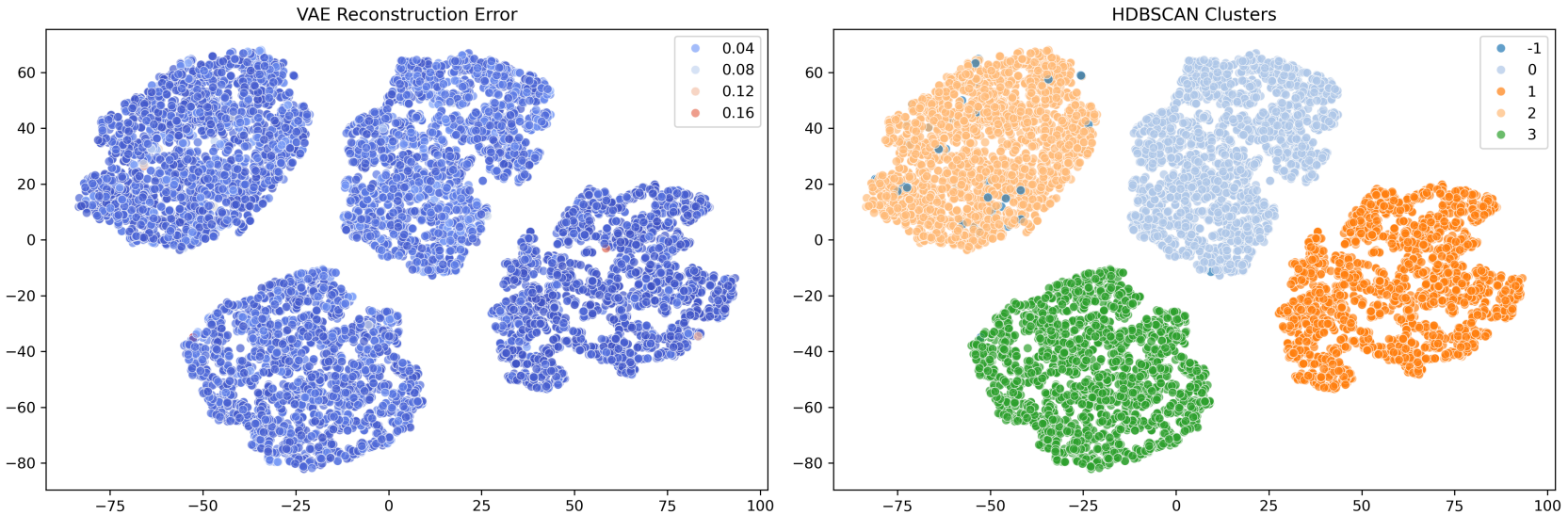
Unsupervised Learning Results



Unsupervised Learning Results for Attack Patterns classification

Unsupervised Learning Results continue

HDBSCAN Clusters plot illustrates the results of Hierarchi- cal Density-Based Spatial Clustering of Applications with Noise (HDBSCAN)



Unsupervised Learning Results for Attack Patterns classification

Performace Results

Classification performance comparison

Model	Precision	Recall	F1-Score
Isolation Forest	0.85	0.82	0.83
VAE	0.89	0.84	0.86
HDBSCAN	0.83	0.80	0.81
Ensemble	**0.92**	**0.88**	**0.90**
SVM Baseline	0.90	0.87	0.89

Conclusion

- Attack-agnostic detection with SHAP-based interpretability and automated countermeasure triggering.
- 92% F1-score gain via temporal feature engineering, ensemble methods, and latent space modeling.

Thank You

Questions and Answers