RUHR-UNIVERSITÄT BOCHUM

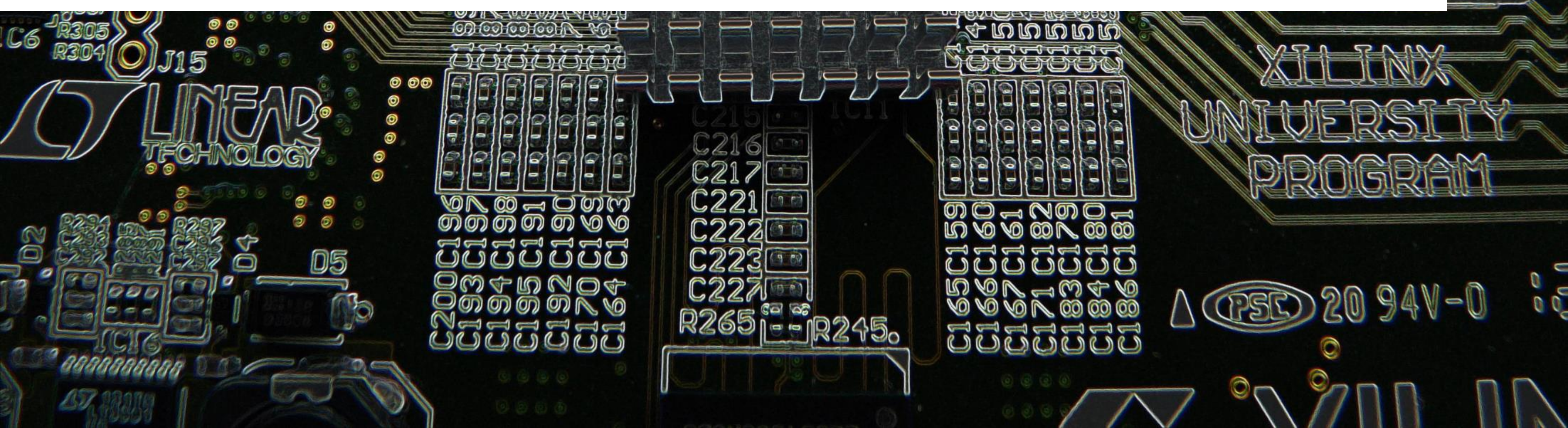
RUB

A Machine Learning Methodology for Cache Recommendation

Osvaldo Navarro, Jones Mori, Javier Hoffmann, Fabian Stuckmann and Michael Hübner

Lehrstuhl für Eingebettete Systeme der Informationstechnik (ESIT)
Fakultät für Elektrotechnik und Informationstechnik
Prof. Dr.-Ing. habil. Michael Hübner

www.esit.rub.de

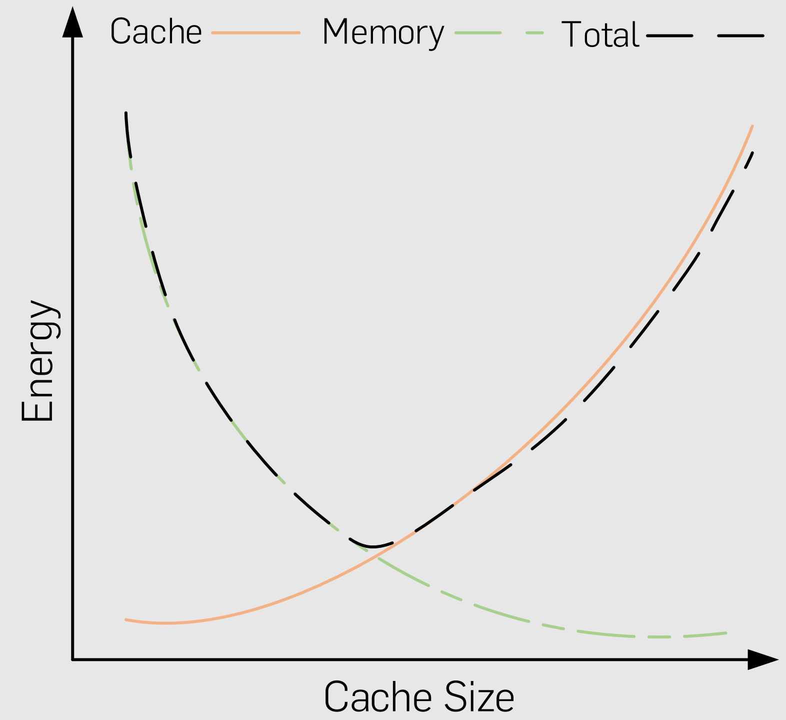


Content

- Introduction
- Supervised Learning
- Cache Recommendation
- Experimental Setup
- Experiments & Results
- Concluding Remarks

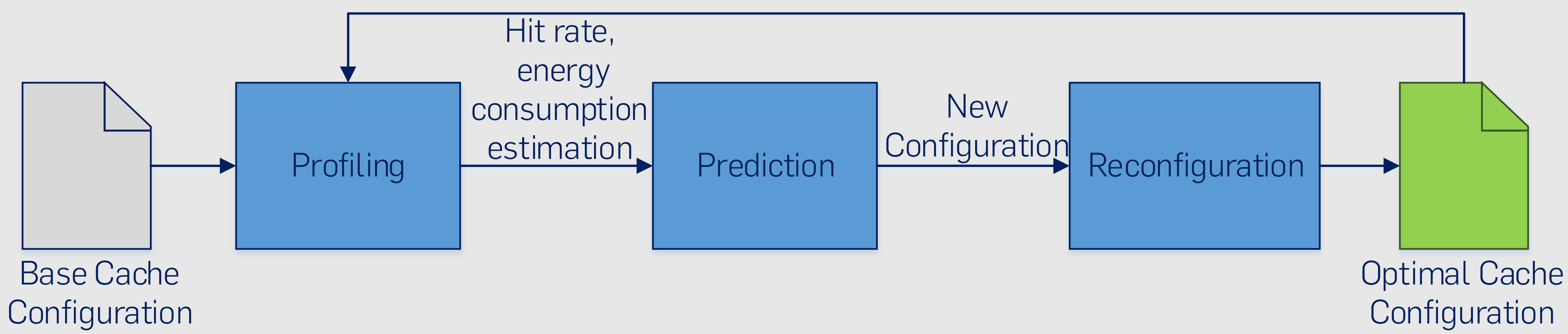
Motivation

- Current designs - *one size fits all*
- Cache consumes a great deal of energy
- Applications benefit from cache adaptability
- Each application might have a different optimal cache configuration
- Within applications, requirements to the cache might change

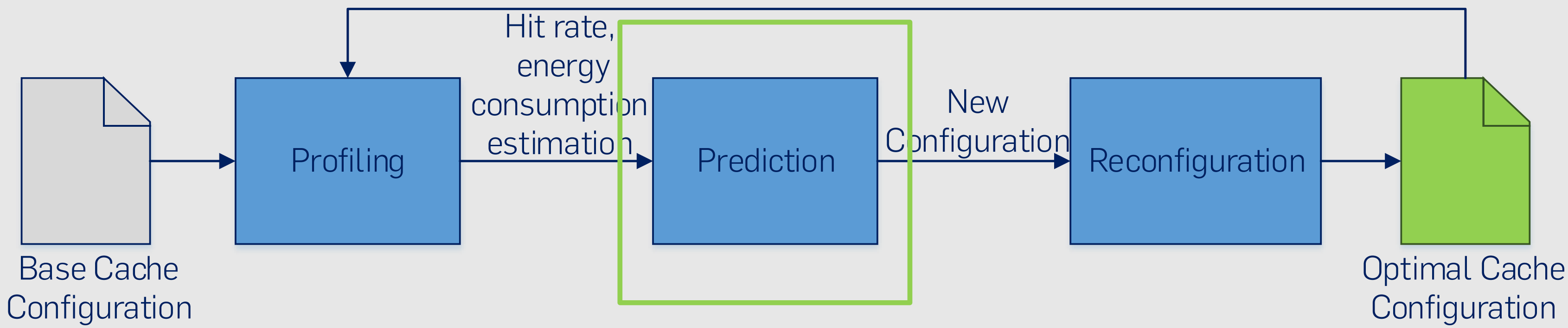


Dynamic Cache Reconfiguration (DRC)

- Reconfigure the cache during runtime
- Fit every application, and to changes within the application later
- Optimal point between energy consumption and performance



Dynamic Cache Reconfiguration (DRC)

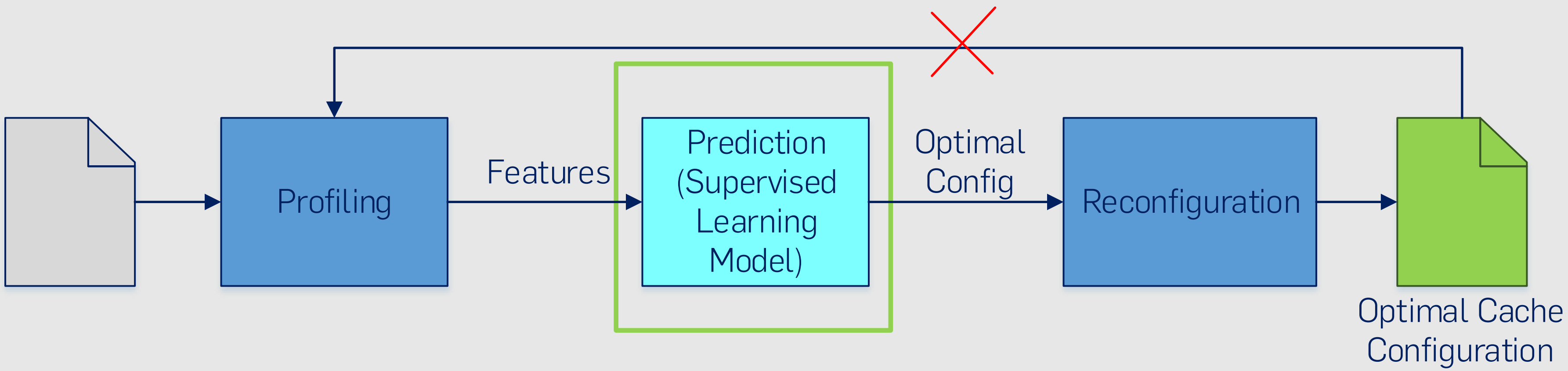


- Run-time algorithms: tune one parameter at a time
- Takes several reconfigurations to converge

Dynamic Way-shutdown algorithm. [1]

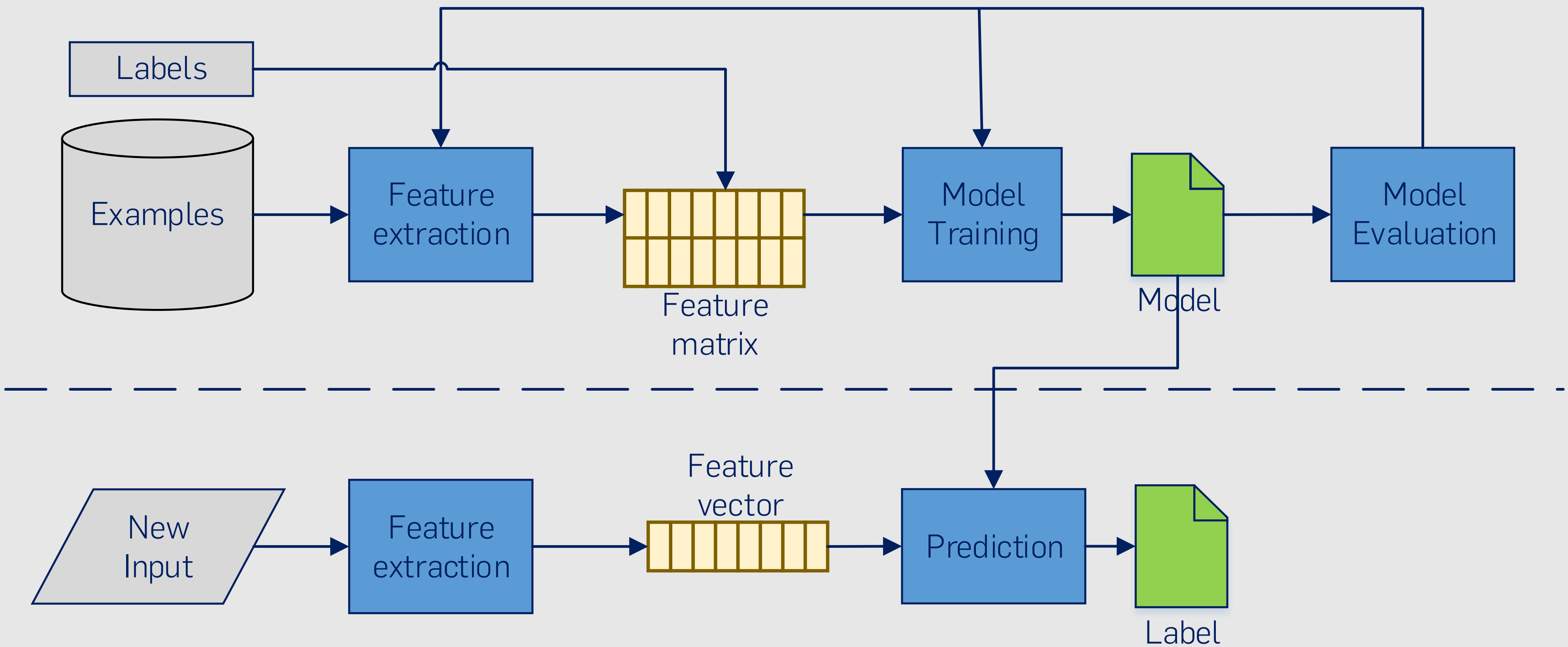
$\delta = MissRate_{current} - MissRate_{previous}$;
if $\delta > 0$ **then**
  if $\delta < \alpha$ **then**
    tolerance++;
    if $tolerance \geq \beta$ **then**
      increaseCacheWays;
      tolerance = 0;
  else
    increaseCacheWays;
    tolerance=0;
else
  tolerance=0;
  decreaseCacheWays;

Cache Recommendation by Supervised Learning



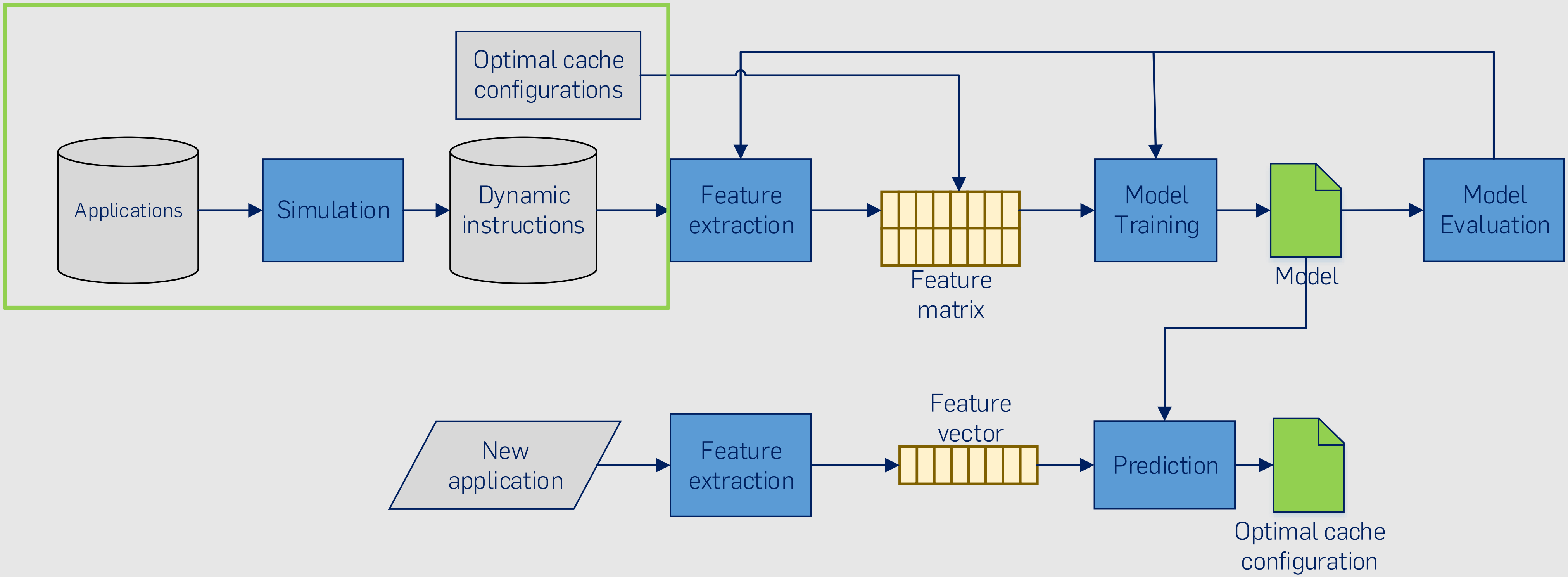
- Using a supervised learning model instead of an iterative algorithm avoids continuously reconfiguring the cache
- The model suggest a final configuration and avoids testing one parameter at a time

Supervised Learning Flow



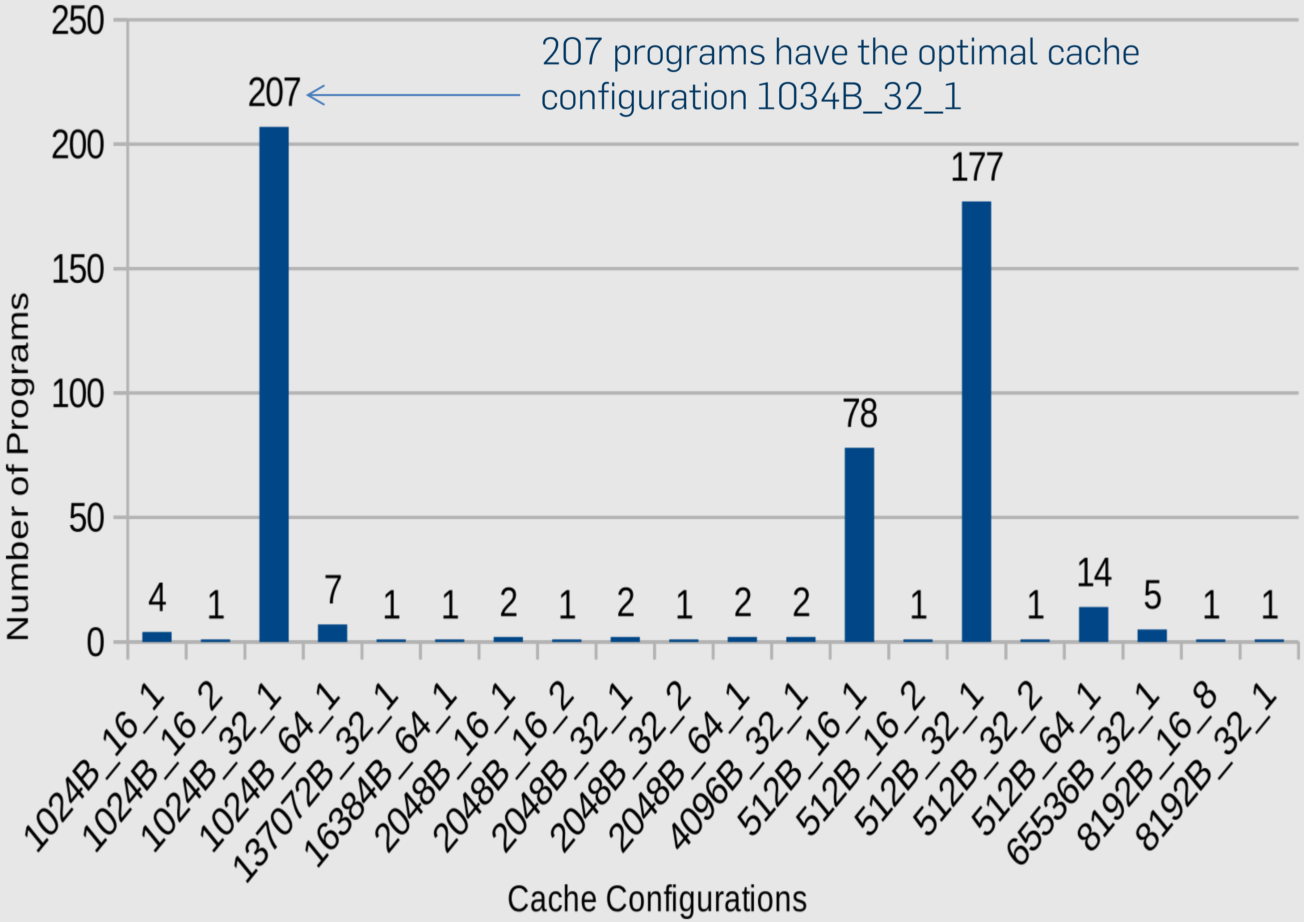
Cache Recommendation Flow

- View the cache prediction problem as supervised learning problem
- Use dynamic instructions as features to represent the programs



Training Set

- Distribution of programs per optimal cache configuration
- Obtained from miBench [4] and Florida State University's Website[5]

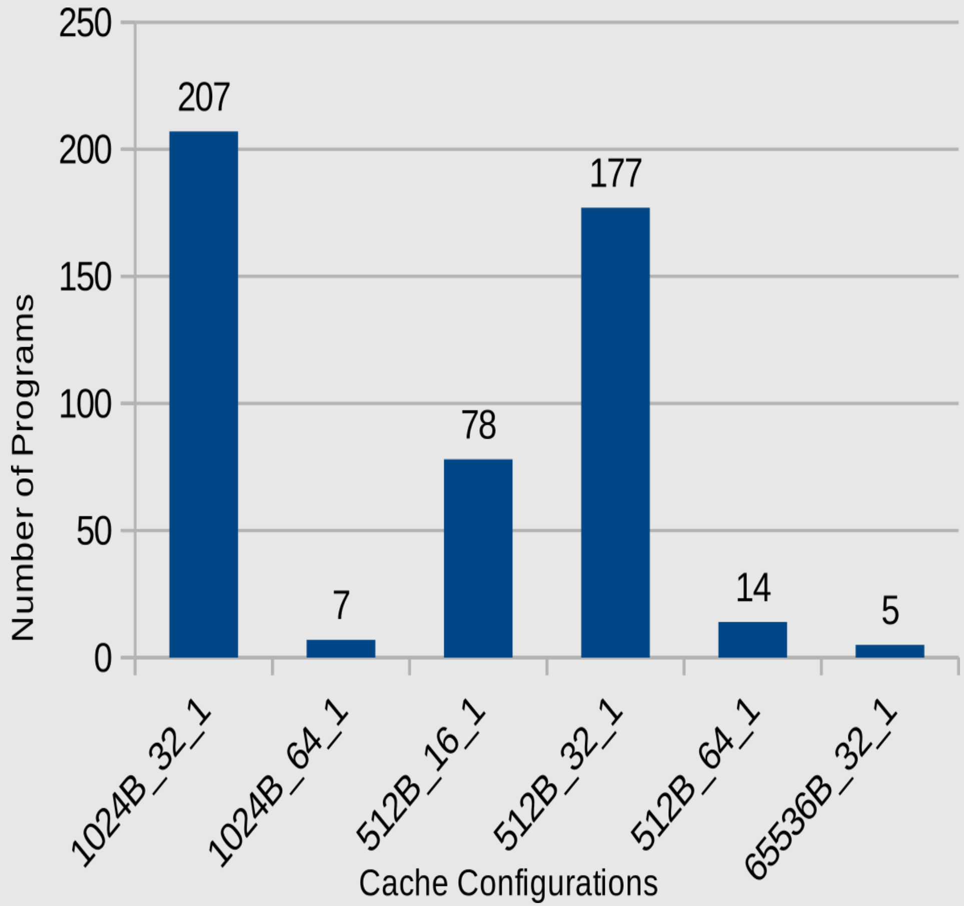


- Cache configuration represented as:

cacheSize_lineSize_associativity

Training Set

- We applied a threshold of 5 applications per cache configuration to focus only in the configurations with enough examples to train the system



- Cache configuration represented as:

cacheSize_lineSize_associativity

Feature Matrix

- We represented each application with the relative frequencies of their dynamic instructions
- A cache configuration consists on: cache size, line size and associativity

From Benchmarks

Application	inst0	inst1	…	instn	Optimal Cache config
App0	freq_0_0	freq_0_1	…	freq_0_n	cc_0
App1	freq_1_0	freq_1_1	…	freq_1_n	cc_1
…					
Appm	freq_m_0	freq_m_1	…	freq_m_n	cc_m
bfs	0.13	0.32	…	0.40	512_16_2

Experimental Setup

- Simulation: Gem5 [1]
  - Input: applications' source code in C
  - Output: number of cache hits, cache misses, etc.

- Energy consumption estimation: CACTI 4.1 [2]
  - Input: cache configuration
  - Output: power estimation statistics

- Supervised learning algorithms: Weka 3.8 [3]
  - Input: feature matrix
  - Output: performance statistics

- Benchmark: 488 applications from miBench [4] and Florida State University's Website[5]

Experimental Setup

- Energy consumption estimation

$$E_{cache} = E_s + E_d$$

$$E_s = cycles * E_{StatPerCycle}$$

$$E_d = E_{hit} * cacheHits + E_{miss} * cacheMisses$$

E_{cache}: cache's energy consumption
E_s: cache's static energy consumption
E_d: cache's dynamic energy consumption
$E_{StatPerCycle}$: static energy per cycle
E_{hit}: energy consumed per cache hit
E_{miss}: energy consumed per cache miss

Experimental Setup

- Evaluation metrics

$$Precision = \frac{tp}{tp+fp}$$

- Percentage of correct positive predictions

$$Recall = \frac{tp}{tp+fn}$$

- Percentage of optimal configurations correctly predicted

$$F_{measure} = 2 * \frac{Precision \; * Recall}{Precision + Recall}$$

- Harmonic mean of precision and recall

tp: number of true positives
fp: number of false positives
fn: number of false negatives

Results

- 68.6% of the applications were assigned their optimal cache configuration

Classifier	Precision %	Recall %	F-Measure
RandomSubSpace	68.6	69.5	0.683
LMT	68.0	68.9	0.682
Simple Logistic	68.0	68.9	0.682
SMO	67.5	69.1	0.681
Multilayer Perceptron	66.3	67.6	0.669

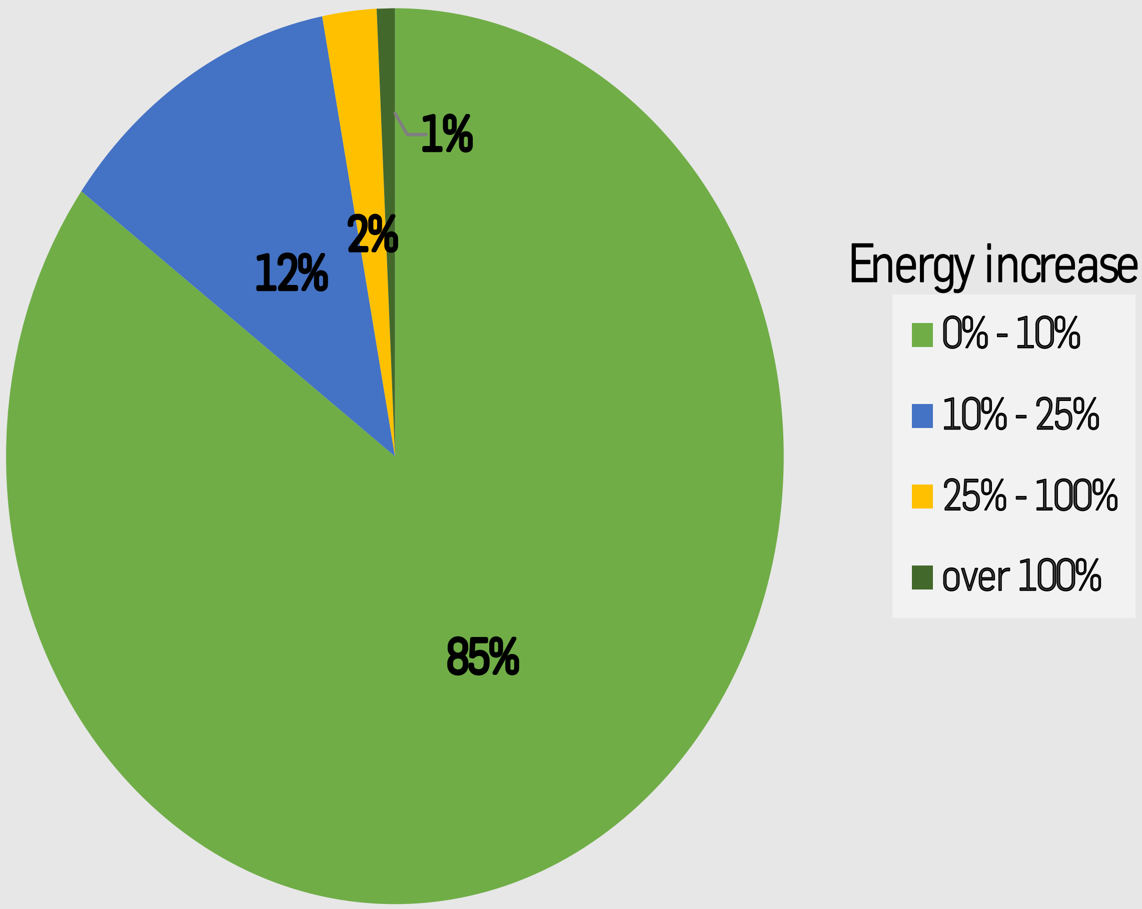
Results – Resampling Filter

- Next, a resample filter was applied to increase the number of instances of the less populated classes (configurations)
- This lead to an significantly improvement of the results

Classifier	Precision %	Recall %	F-Measure
Randomizable FilterClassifier	99.52	79.74	0.866
RandomCommittee	99.47	79.94	0.864
IBK	99.67	78.51	0.858
LMT	98.46	78.10	0.848
RandomForest	99.80	77.68	0.843

Energy Consumption Increase

- The misclassified applications were assigned an almost optimal cache
- 85% of the misclassified applications obtained a cache that consumes only up to 10% energy



Misclassified applications

Concluding Remarks

- **Cache recommendation** methodology based on Machine Learning using dynamic instruction' frequencies as features
- **Up to 99.80 % precision**
- **Finds optimal configuration in fewer iterations** than [1], only needs 1 iteration and 1 reconfiguration
- **Future work: extend database, online implementation**

- Thanks for your attention!

Contact:

M. Sc. Osvaldo Navarro
Chair for Embedded Systems of InformationTechnology
Ruhr-Universität Bochum
Tel: +49 (0)234-32-25950
E-Mail: Osvaldo.NavarroGuzman@rub.de

References

[1] Silva, Bruno A., et al. "**Run-time Cache Configuration for the LEON-3 Embedded Processor**." Proceedings of the 28th Symposium on Integrated Circuits and Systems Design. ACM, 2015.
[1] Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S., et al.: **The gem5 simulator**. ACM SIGARCH Computer Architecture News 39(2), 1{7 (2011)
[2]Tarjan, D., Thoziyoor, S., Jouppi, N.P.: **Cacti 4.0.** Tech. rep., Technical Report HPL-2006-86, HP Laboratories Palo Alto (2006)
[3] **Weka's resample lter**, http://weka.sourceforge.net/doc.dev/weka/filters/supervised/instance/Resample.html
[4]Guthaus, M., Ringenberg, J., Ernst, D., Austin, T., Mudge, T., Brown, R.: **MiBench: A free, commercially representative embedded benchmark suite**. Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538) pp. 3{14 (2001)
[5] **C source codes benchmark, http://people.sc.fsu.edu/~jburkardt/c_src/c_src.html**

Supervised Learning Algorithms

- Several classifiers from WEKA where tested

Type	Algorithm
Bayes	BayesNet, Naive Bayes, Naive Bayes Multinomial
Functions	Multilayer Perceptron, Simple Logistic, SMO
Lazy	LBK, LWL, KStar
Meta	AdaBoostM1, Attribute Selected Classifier, Bagging, CV Parameter Selection, Filtered Classier, Iterative Classier Optimizer, LogitBoost, Multiclass Classier, Multischeme, Random Committee, Random Subspace, Randomizable Filtered Classier, Stacking, Vote
Misc	Input Mapped Classifier
Rules	Decision Table, JRip, PART, OneR, ZeroR
Trees	Decision Stump, Hoeding Tree, J48, LMT, Random Forest, Random Tree, REPTree