Institut de Recherche en Informatique et Systèmes Aléatoires

Machine Learning for Timing Estimation: the good, the bad and the ugly – short version

Isabelle Puaut

RT-ML workshop, 2024

Université de Rennes

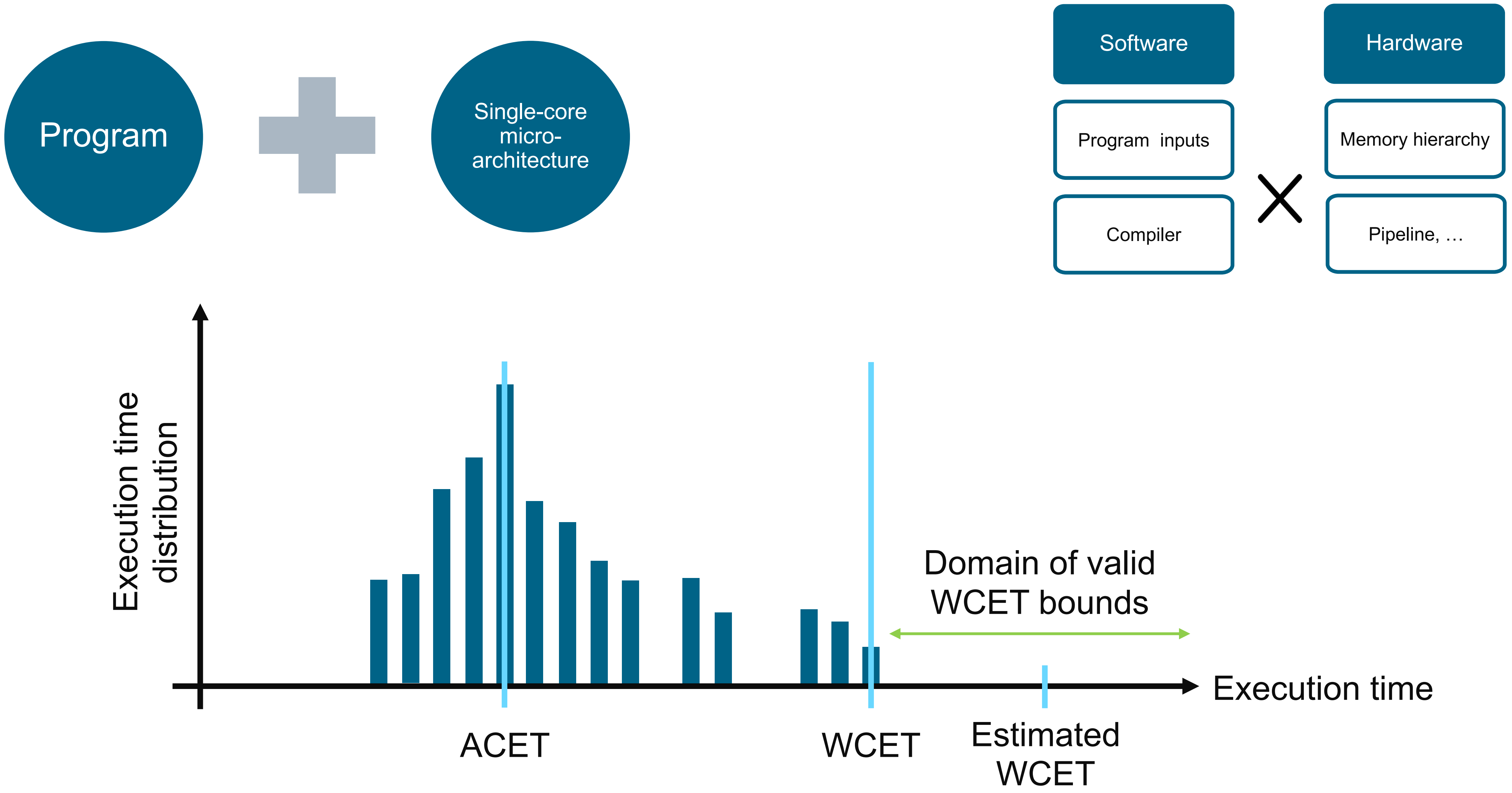
IRISA

Inria

Outline

- Motivations for using ML in WCET analysis
- Our contributions (in a nutshell)
- The good
- The bad
- The ugly
- Takeways

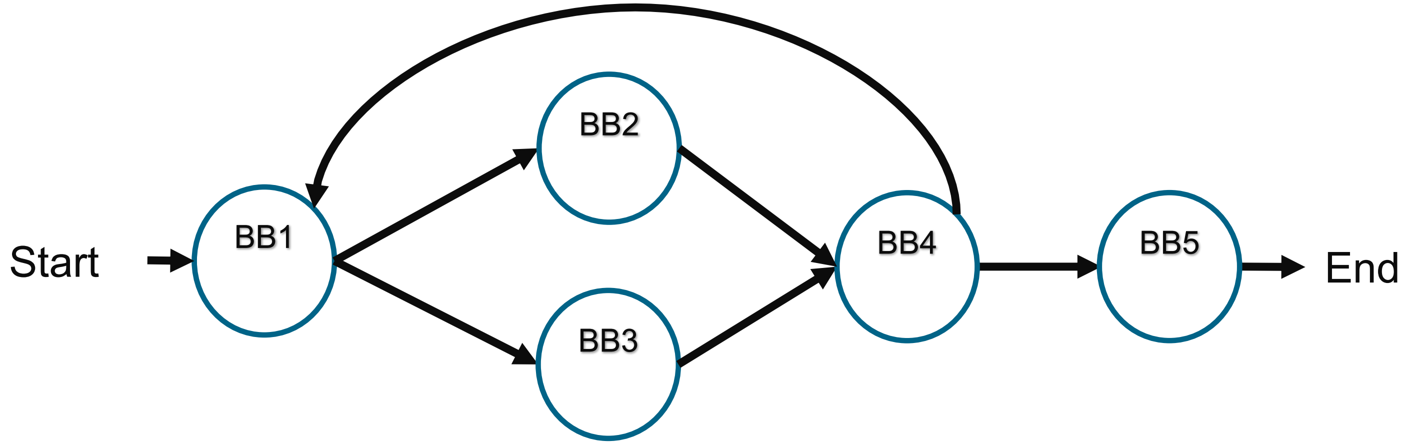
Complexity of WCET estimation



WCET estimation using static timing analysis

- Flow determination (e.g. loop bounds)
- Low-level analysis (hardware model)
- WCET calculation (e.g. IPET)

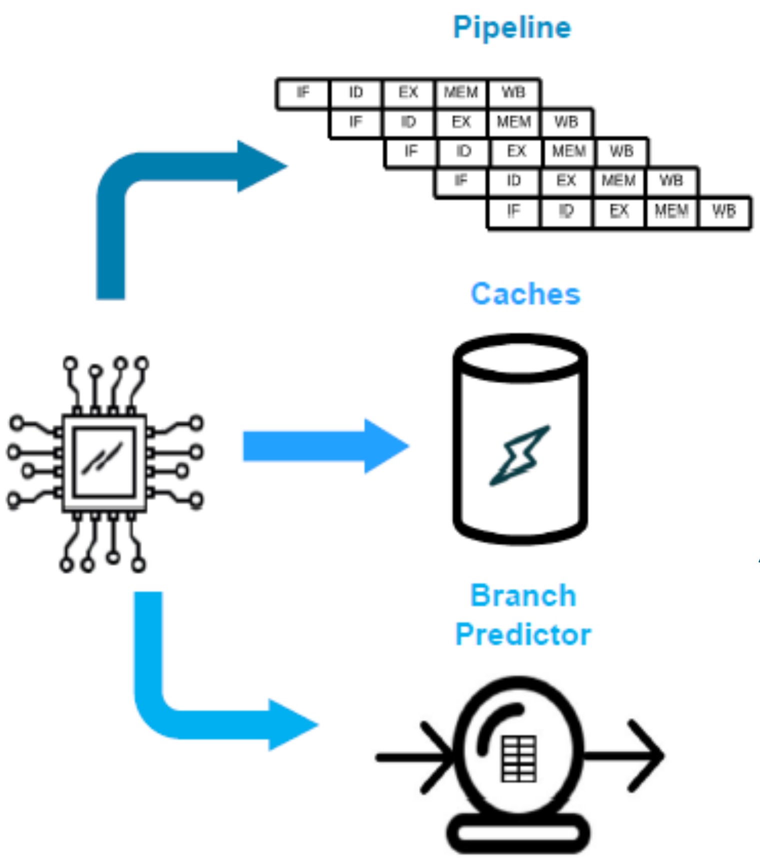
Control Flow Graph (CFG)

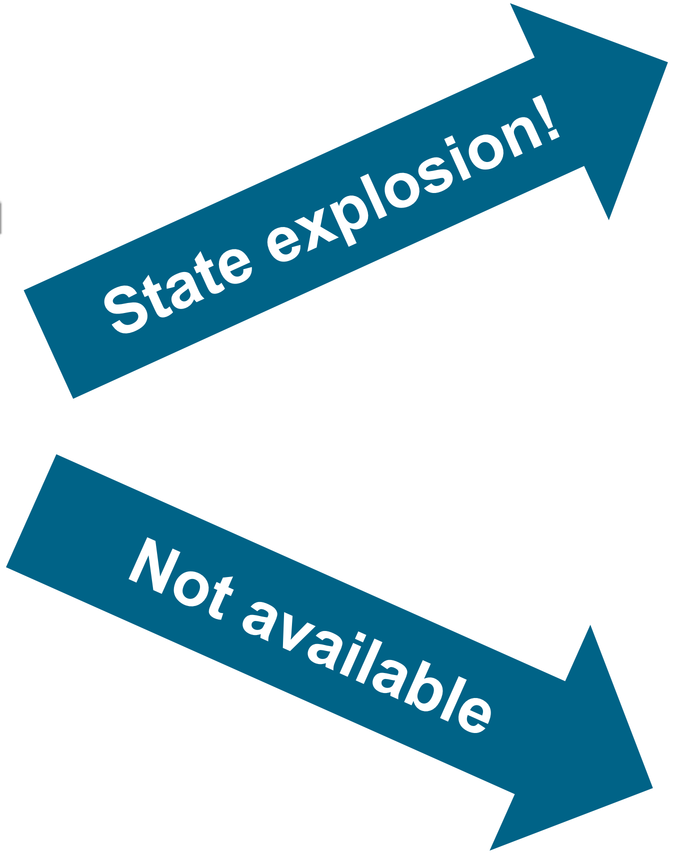


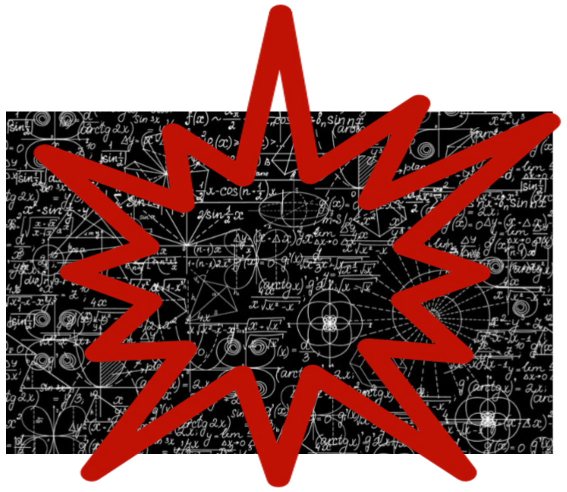
Basic Block (BB): straight-line code sequence with no branch except last instruction

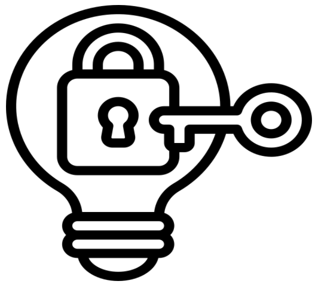
WCET static analysis: limits of low-level analysis

Low level analysis on complex hardware









RTL
Cycle accurate model

Contributions

Machine learning for timing estimation

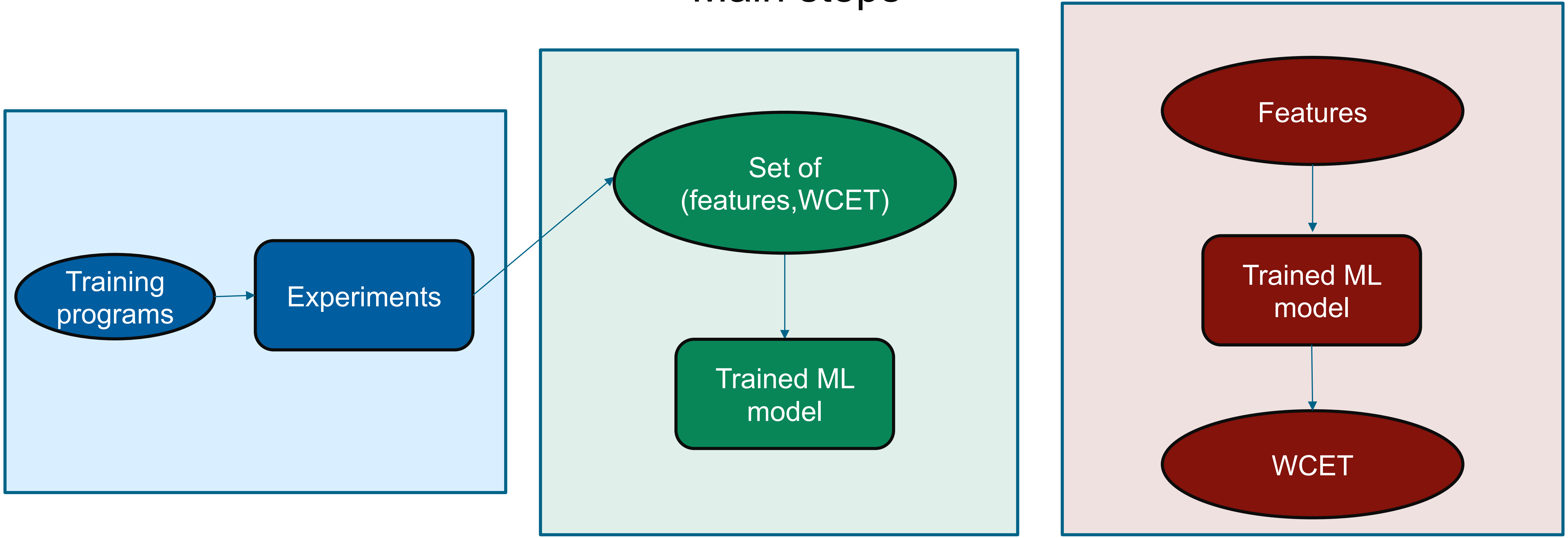
Replace the low-level analysis by a machine learning (ML) model

Supervised learning

Basic block → Machine learning (ML) → WCET

Machine learning for timing estimation

Main steps



1. Collecting training data

2. Training the model

3. Prediction

Spectrum of contributions

- Collection of timing data (at basic block level)
  - Synthetic programs vs real code
  - Metric: average-case performance and worst-case timing
  - Features: proportions of instructions, sequences of instructions (BB “in context”)
- Learning
  - Basic techniques: Linear Regression, Random Forests, Gradient Boosting, Neural Networks (scikit-learn)
  - Natural Language Processing (NLP) Techniques: LSTM, Transformers-XL
- Large panel of architectures:
  - Very simple ones: TI MSP430, Cortex M4
  - More complex ones: Cortex M7, Cortex A53

CAWET: Context-Aware Worst-Case Execution Time Estimation Using Transformers (ECRTS 2023)

Use of Natural Language Processing models: binary code as natural language

operation/operands = letters	instruction = word	basic block = sentence	basic block sequence = paragraph
mov r3 #0	mov r3, #0	mov r3, #0 str r3, [fp, #-12] b .L3	BB1 BB2 BB3

CAWET: use of Transformers XL

Deep learning model
for processing (very) long sequential data

[13] DAI Z., et al. Transformer-xl: Attentive language models beyond a fixed-length context 2019

The good

Benefits of machine learning for timing estimation (by construction)

No need for details of the processor microarchitecture	Once deployed, no measurements needed	Easy porting to a new architecture	Tokenization for free
Only need to measure (in the worst-case)	Fast predictions	Only re-train	Tokenizers exist

Good precision with simple targets

ML algorithm	MAPE (MSP430)
Linear regression	56.7%
Bayesian Ridge	62.1%
Gradient Boosting	42.1%
Random Forest	43.4%
Multilayer perceptron	**8.2%**

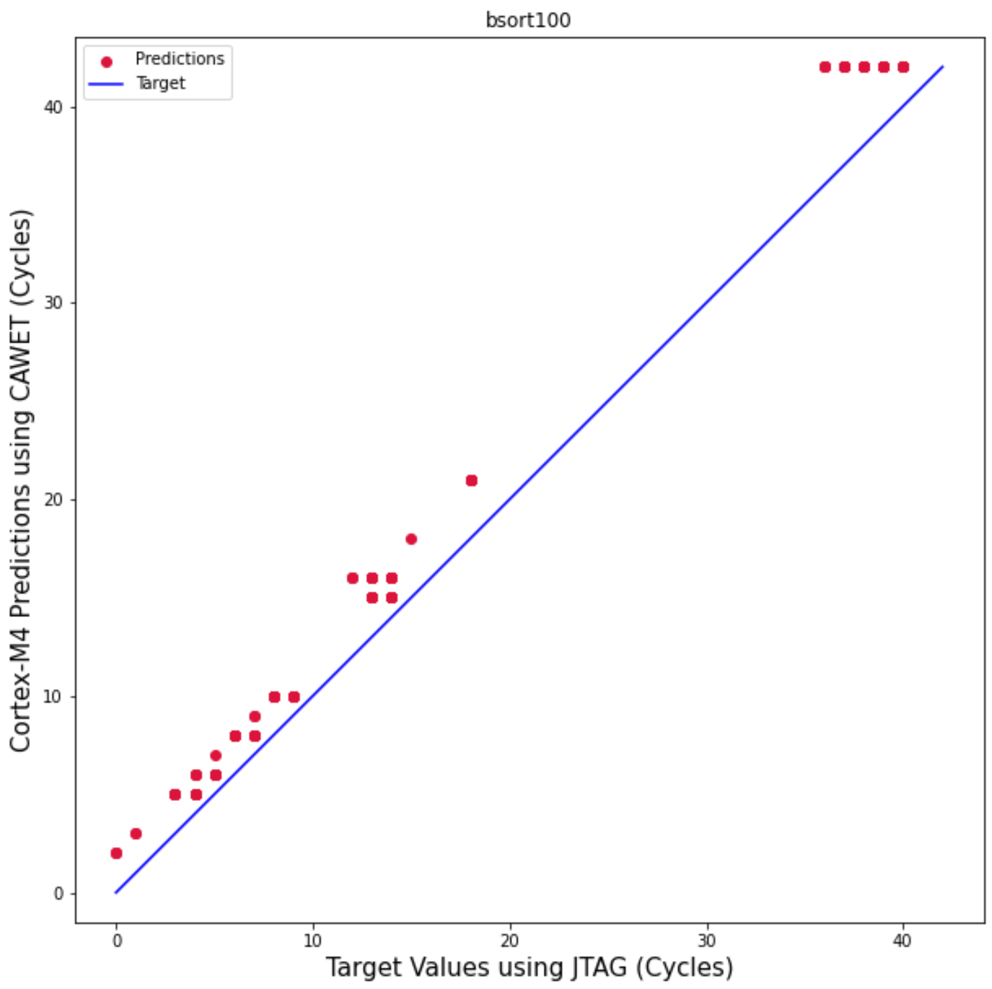
Low-power TI MSP430 micro-controller (2-stage pipeline, tiny icache, no dcache), basic ML algorithms

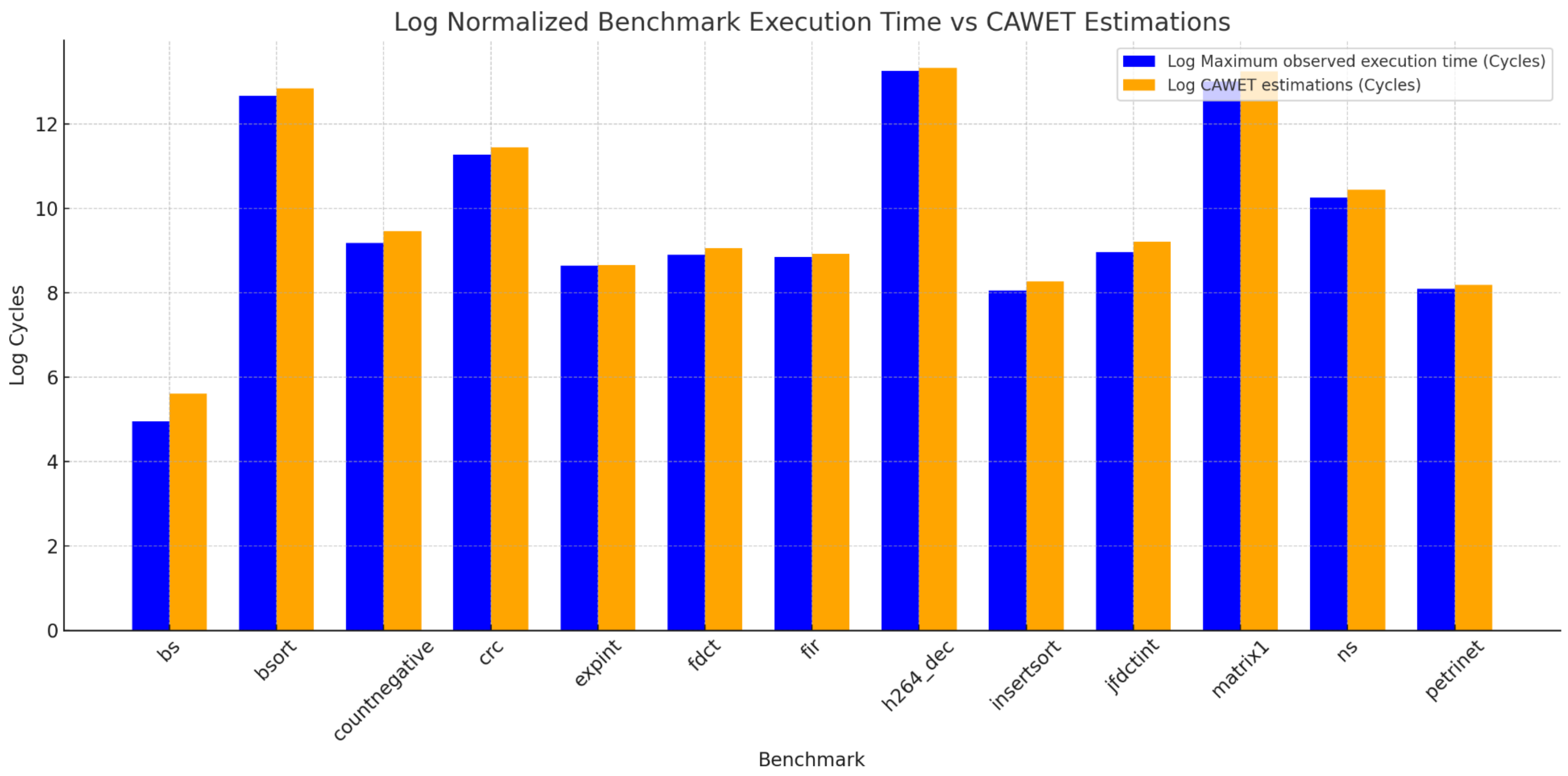
Still acceptable precision with a more complex target

ML algorithm	MAPE (Cortex M4)
Multilayer perceptron	43.8%
LSTM	36.2%
CAWET (Transformer-XL)	**23.8%**

MAPE on Cortex M4 (in-order pipeline, 3-stages, no cache, jtag)

Never under-estimates WCETs





On Cortex M4 (BB level on left, program level at right)

The bad

Pessimism augments with more complex targets

ML algorithm	MAPE (Cortex M4)	MAPE (Cortex M7)
Multilayer perceptron	43.8%	132.7%
LSTM	36.2%	126.4%
CAWET (Transformer-XL)	23.8%	102.2%

Cortex M7 (in-order pipeline, 6-stages, L1 caches, jtag)

Hyper-parameter selection may get you nuts

Loss function	MSE						MAPE					
Learning rate	10^{-4}		10^{-3}		10^{-2}		10^{-4}		10^{-3}		10^{-2}	
Optimizer	SGD	ADAM	SGD	ADAM	SGD	ADAM	SGD	ADAM	SGD	ADAM	SGD	ADAM
Default	163%	159%	182%	198%	170%	195%	152%	161%	210%	176%	182%	169%
Larger ML network size	210%	139%	156%	167%	321%	124%	110%	134%	152%	143%	126%	134%
Without float instructions	77%	69%	88%	65%	75%	98%	78%	65%	56%	55%	89%	60%
Learning on normalized time	19.2%	18.7%	19.5%	19.2%	17%	22.1%	15.2%	**11.4%**	14.4%	15.9%	17.3%	19,1%

ACET learning, LSTM, Cortex-M7, learning lasts several days

Feature selection is tricky

- Handcrafted features, Multi-Layer-Perceptron (MLP)

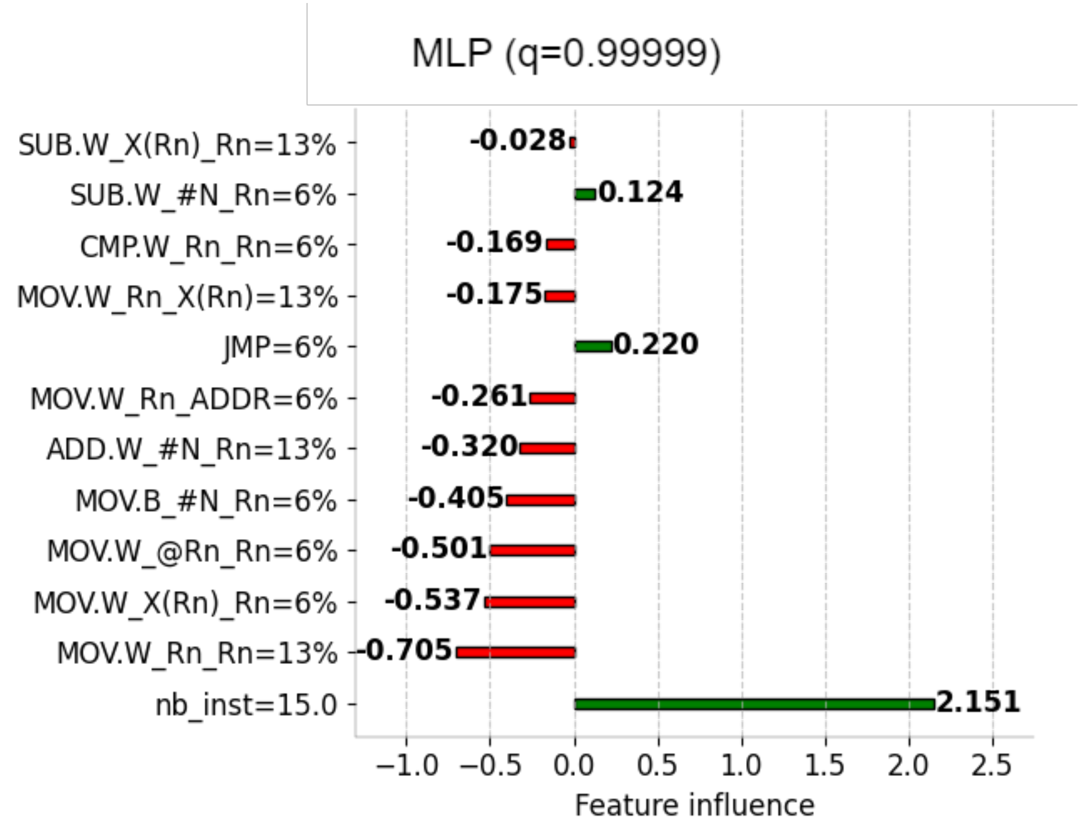
Instruction proportions	Adding access type
%MOV, %ADD, %SUB…	%instruction_with_direct_access %instruction_with_indirect_access…
Error = 311%	Error = 181%

Models are hard to debug

Instruction_Src_Dst	MOV.W_Rn_Rn	MOV.W_@Rn_Rn	MOV.B_#N_Rn	MOV.W_X(Rn)_Rn	ADD.W_#N_Rn	CMP.W_Rn_Rn	MOV.W_Rn_X(Rn)	SUB.W_#N_Rn	MOV.W_Rn_ADDR	JMP	SUB.W_X(Rn)_Rn

Is it correct?

12 cycles

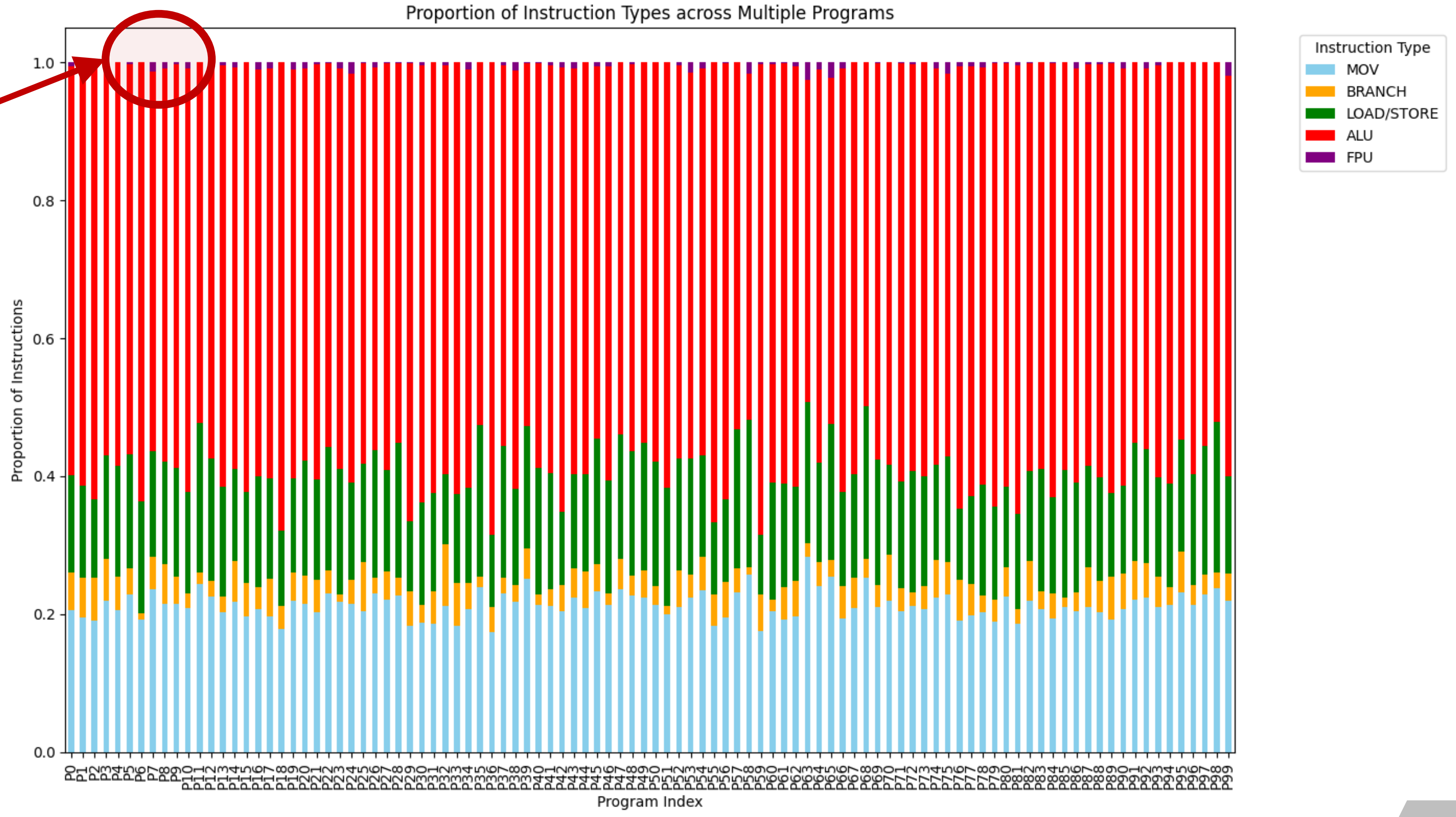


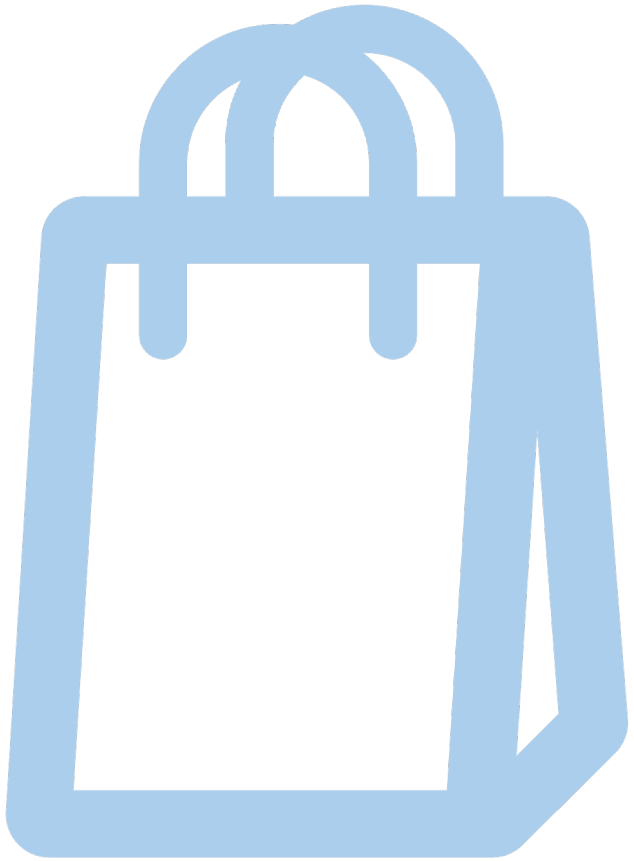
- Local Interpretable Model-agnostic Explanations (LIME) - see Wortex talk
- The most impacting feature is instruction count, really?

The ugly

Rare features may destroy all

- Float instructions have highly variable timing
- VDIV in Cortex-A53 can take between 4 and 32 cycles
- In CAWET Cortex-M7, we gain ~200% error reduction when removing them





Takeaways

Takeways: lessons learnt

- Feature selection is key to success
- Training data is crucial
- ML for timing estimation works pretty well, but
  - Many (too many) parameters to control: be calm, patient, and methodic
  - Requires domain expertise and ML expertise: cooperate with ML experts!
  - Techniques hard to debug: need for (more) explainability
  - No formal guarantee of safety/precision: certifiable ML

Takeways: future work

- Extension multi-cores

Postdoc position (University of Toulouse, and University of Rennes, France), project AIxIA (Artificial Intelligence for Interference Analysis)

For the bounty please contact:

- Thomas CARLE : thomas.carle@irit.fr
- Isabelle PUAUT : puaut@irisa.fr

- More details needed? Join the WCET workshop for the long version!

Any question?

No question, really?

Publications

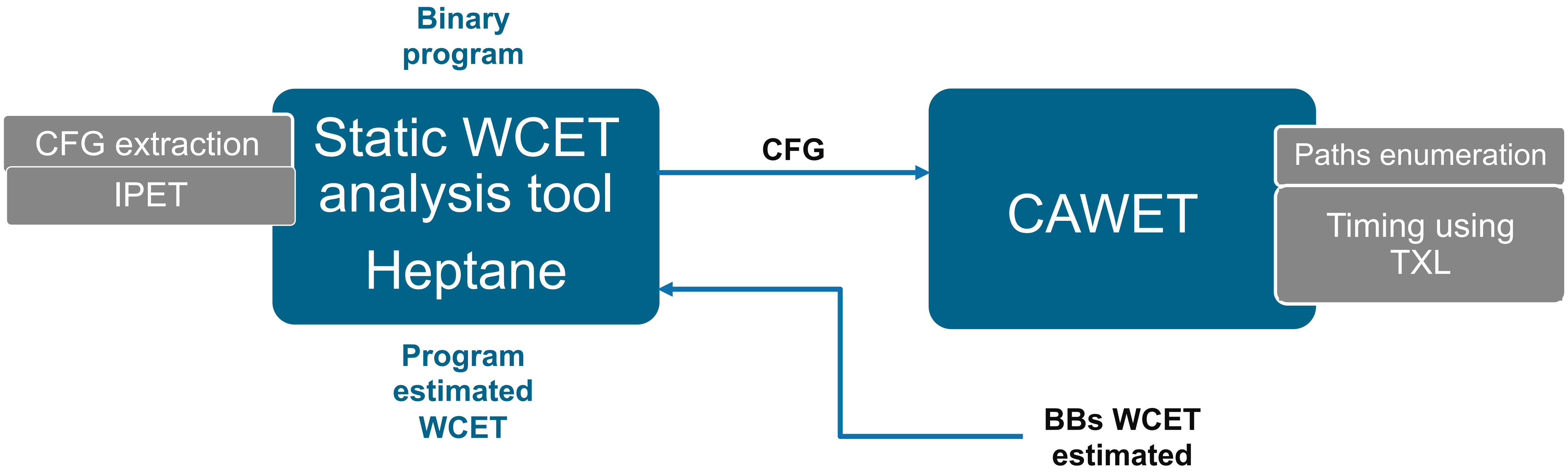
- A. N. Amalou, I. Puaut and G. Muller. WE-HML: Hybrid WCET Estimation using Machine Learning for Architectures With Caches. RTCSA 2021.
- A. N. Amalou, E. Fromont and I. Puaut. CATREEN: Context-Aware Code Timing Estimation with Stacked Recurrent Networks. ICTAI 2022.
- A. N. Amalou, E. Fromont and I. Puaut. CAWET: Context-Aware Worst-Case Execution Time Estimation Using Transformers." ECRTS 2023.
- A. N. Amalou, E. Fromont and I. Puaut. Fast and Accurate Context-Aware Basic Block Timing Prediction using Transformers.". CC, 2024.
- H. Reymond, A. N. Amalou and I. Puaut. WORTEX: Worst-Case Execution Time and Energy Estimation in Low-Power Microprocessors using Explainable ML. WCET 2024

Datasets

- A. N. Amalou, I. Puaut. A dataset of synthetically generated code blocks for the learning of WCET on Cortex A53 [Dataset]. Zenodo
- A. N. Amalou, E. Fromont and I. Puaut. Training dataset for transformers consisting of basic blocks and their execution times along with the execution context of these blocks, for various Cortex processors M7, M4, A53, and A72. [Dataset]. Zenodo.
- H. Reymond, H. Chabot, A. N. Amalou, I. Puaut, MSP430FR5969 Basic Block Worst-Case Energy Consumption (WCEC) and Worst-Case Execution Time (WCET) dataset. [Dataset]. Zenodo

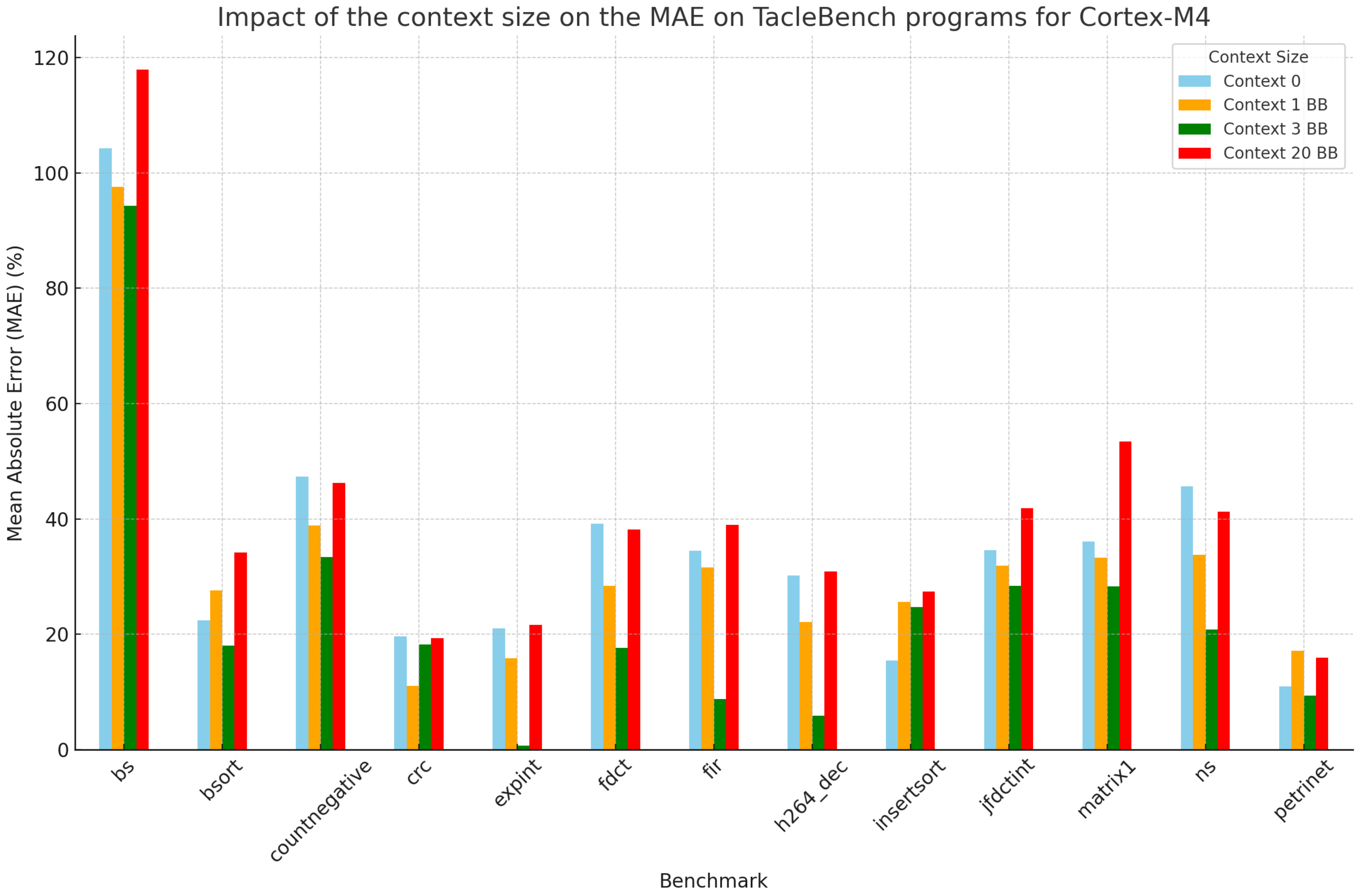
CAWET: use of Transformers XL

Details on prediction

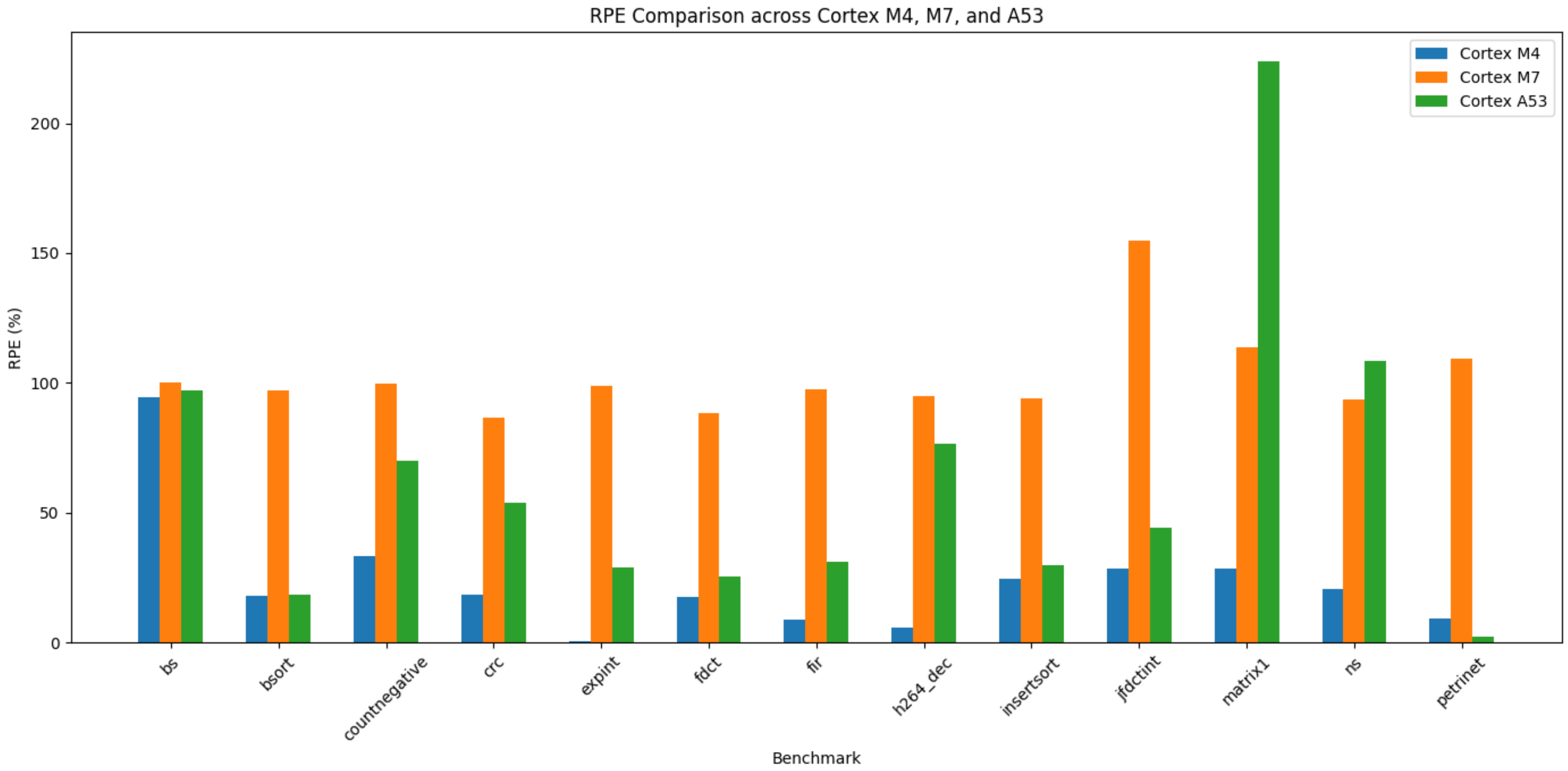


[22] Hardy, D., Rouxel, B., & Puaut, I. (2017). The heptane static worst-case execution time estimation tool. In *17th International Workshop on Worst-Case Execution Time Analysis*

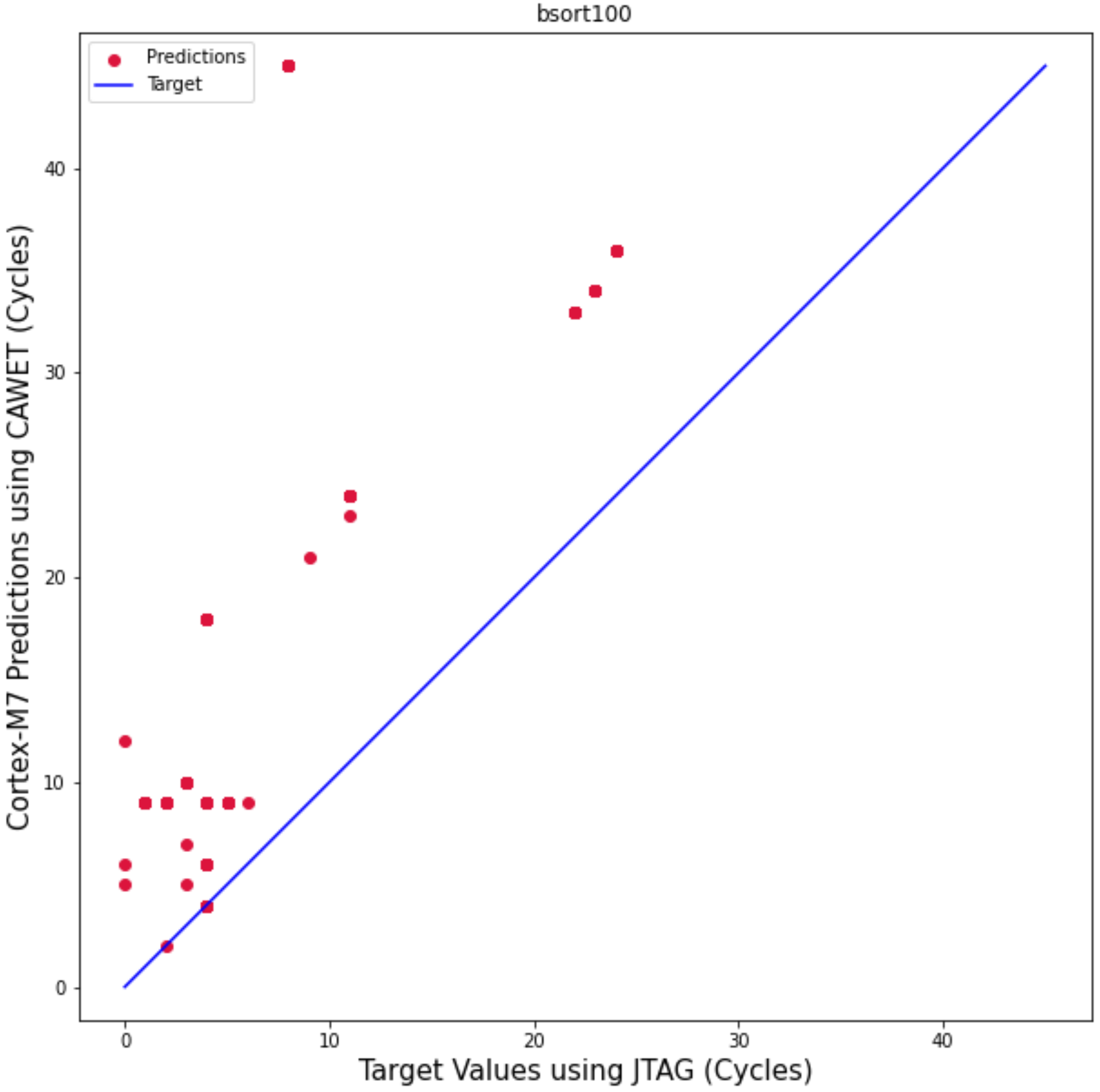
Context size

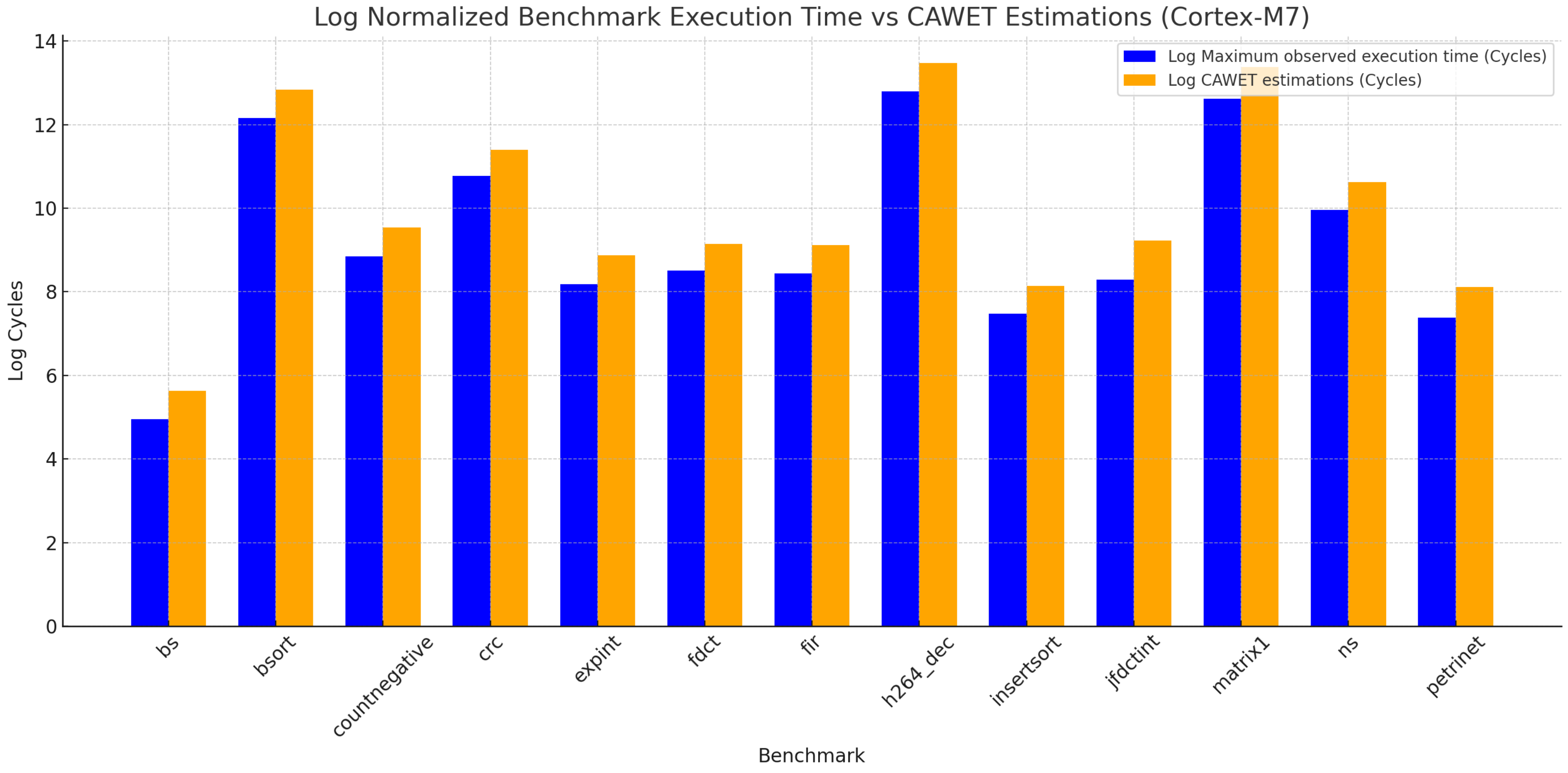


Pessimism and hardware complexity



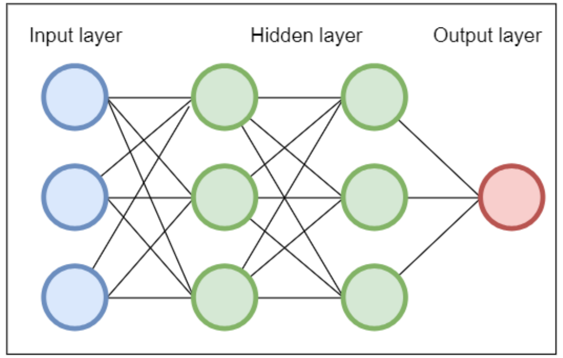
Never under-estimates WCETs (Cortex-M7)



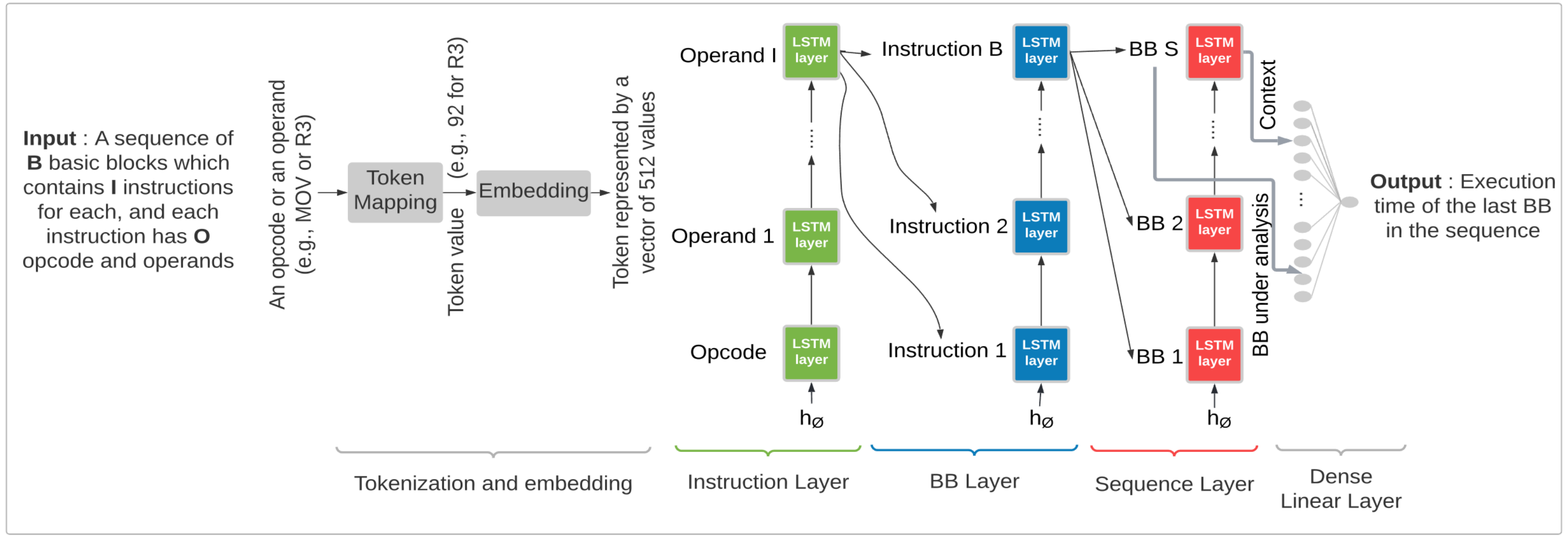


Experimental setup: competitors

Multilayer perceptron (MLP) based on the work of WE-HML [8]



LSTM based: ITHEMAL [20] and CATREEN [9]



[8] AMALOU A. N., PUAUT I. and MULLER G. "WE-HML: Hybrid WCET Estimation using Machine Learning for Architectures With Caches." The 27th International Conference on Embedded and Real-Time Computing Systems and Applications. IEEE, 2021.

[20] MENDIS, C., et al. Ithemal: Accurate, Portable and Fast Basic Block Throughput Estimation using Deep Neural Networks. International Conference on Learning Representations, 2018.

[9] AMALOU A. N., FROMONT E., and PUAUT I. "CATREEN: Context-Aware Code Timing Estimation with Stacked Recurrent Networks." The 34th IEEE International Conference on Tools with Artificial Intelligence IEEE, 2022.

ARM targets used in experimental evaluation

Core	Cortex-M4	Cortex-M7	Cortex-A53
Board	STM32F407	STM32H743	Raspberry Pi 3
Pipeline type & (#stage)	In-order (3)	In-order superscalar (6)	In-order superscalar (8)
Cache memory	N/A	L1	L2
Branch predictor	N/A	Yes	Yes
Measurement solution	JTAG	JTAG	Instrumentation
Microarchitecture complexity	Low	Medium	High

ORXESTRA (ACET) on M4, M7, A53. Evaluation metrics: (MAPE)

Target	Metric	MLP [1]	ITHEMAL [9]	CATREEN [2]	ORXESTRA [3]
Cortex-M4	MAPE	26.4%	14.4%	8.8%	**7.8%**
Cortex-M7	MAPE	22.7%	17.6%	13.3%	**9.6%**
Cortex-A53	MAPE	38.4%	10.1%	8.5%	**5.2%**

- ORXESTRA outperforms all models on all targets
- Context aware models are better than context-agnostic ones

CAWET: Context extraction

Extract for each BB its execution context, constituted by a selected number of previously executed BB

Exhaustive search in a CFG = Path explosion $O(2^{\#bb})$

Solution:

- Divide to conquer
- Local exploration in SESE regions
- (more details if asked for)

[21] DEGIOANNI T., & PUAUT I. StAMP: Static Analysis of Memory access Profiles for real-time tasks. In 20th International Workshop on Worst-Case Execution Time Analysis 2022.

Comparison of WCET predictions for CAWET [12], WE-HML [8] and a neural network baseline on TacleBench programs

Predictor	Cortex-M4 MRPE	Cortex-M7 MRPE	Cortex-A53 MRPE
WE-HML [1] (Multilayer perceptron)	-	-	494.2%
Multilayer perceptron	43.8%	132.7%	85.7%
CAWET [3]	**23.8%**	**102.2%**	**62.4%**

- CAWET is less pessimistic than WE-HML

Execution context

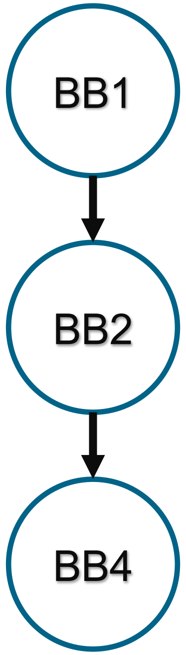
```
int main() {
    int a = 12; int b = 7;
    int max = 0;
    if (a > b) {
        max=a;
    }
    else {
        max=b;
    }
    return max;
}
```

BB1

BB2

BB3

BB4



Context

=

previously executed BBs

+

the basic block itself