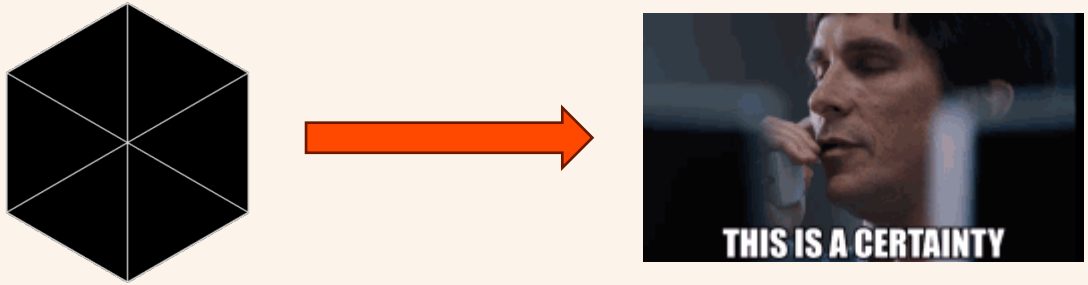
COMPSAC 2025

A Conformal Prediction-Based Framework for CPU Load Forecasting: A Black-Box Approach

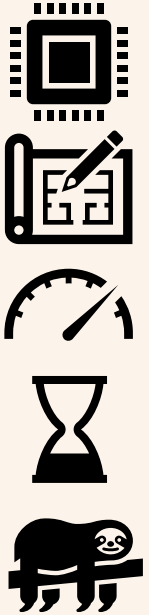
Edin Jelačić, Cristina Seceleanu, Peter Backeman, Ning Xiong, Tiberiu Seceleanu, Axel Jantsch



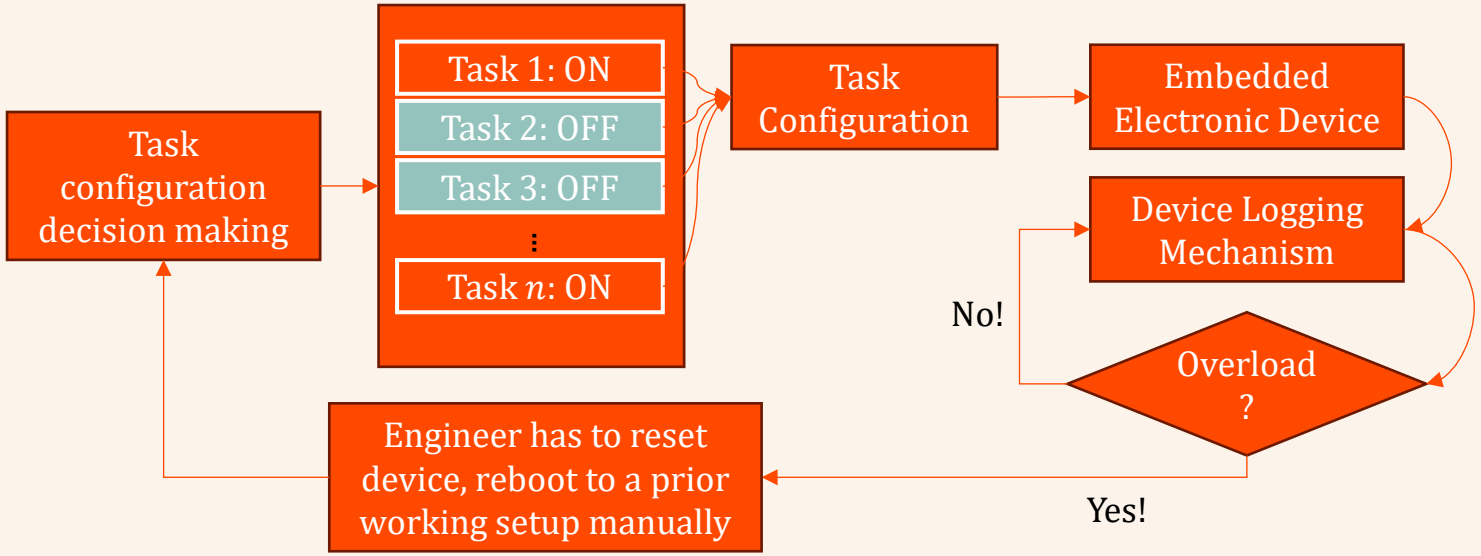
Mälardalen University

The PROBLEM

- Real-time electronic devices in mission-critical applications are constrained by hardware to the number and comb‘s of tasks
- Customers choose their own **tasks**
- Selecting the tasks improperly may lead to too much **CPU** loading
- Overloading the **CPU** (or **CPUs**) leads to a breakdown in **scheduling**
- Machine may **NO LONGER** be real-time!

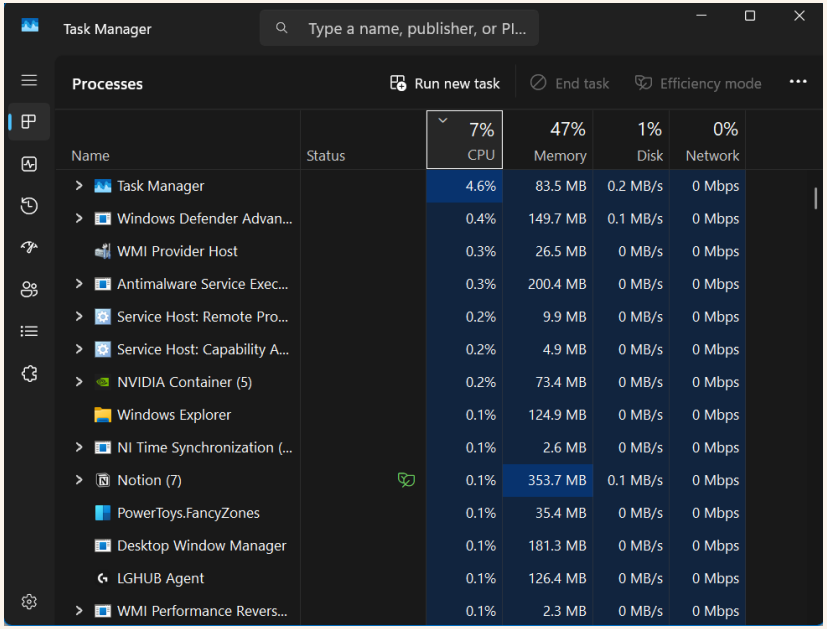


State-of-Practice



The QUESTION

Could we predict the loading on embedded electronic device CPUs ahead of running these tasks, and if so, could this be done with **quantifiable uncertainty**?



Related Work

- Paper on demistifying the black-box nature of latency estimations of ML accelerators with conformal prediction [M. Wess et al., 2024]
- Conformal prediction (CP) shown to be an extremely powerful tool in recent years for statistically sound* uncertainty quantification [V. Manokhin, 2022]

* = under assumption of data exchangeability

Related Work

Shapley value analysis has already been augmented by CP [D. Watson, et al., 2023]

* = under assumption of data exchangeability

The IDEA

Why not do it the other way around?

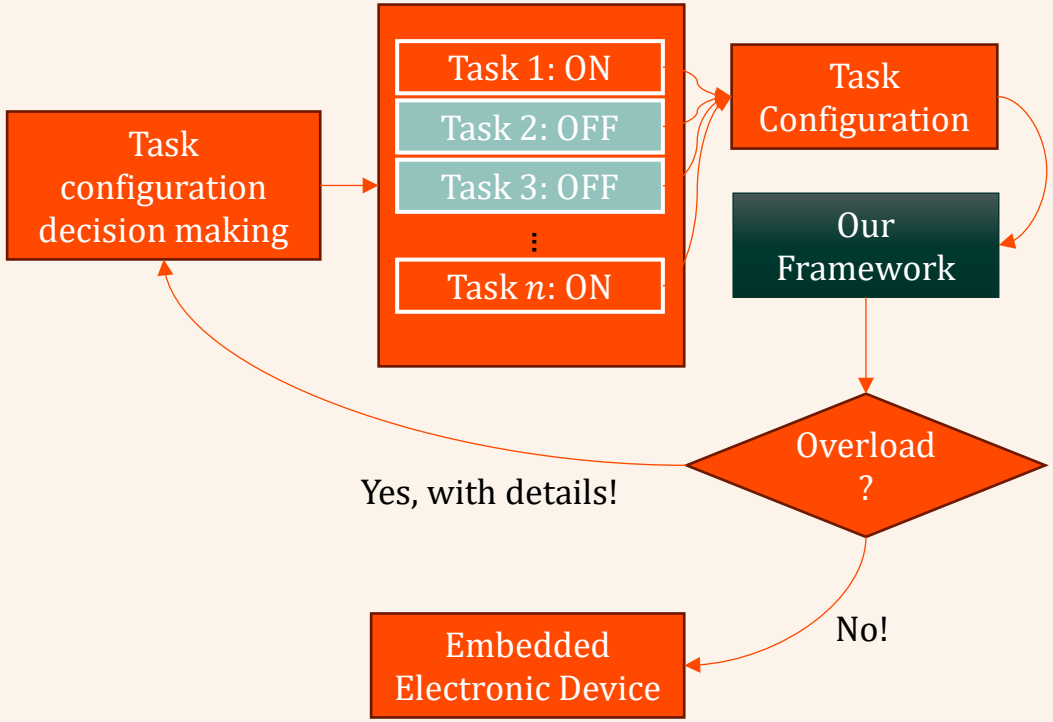
CP augmented by Shapley analysis!

* = under assumption of data exchangeability

The (proposed) SOLUTION

Framework for **Conformal Prediction** of CPU load utilization with **Shapley analysis** for individual **task** attribution

...with an accompanying desktop app!

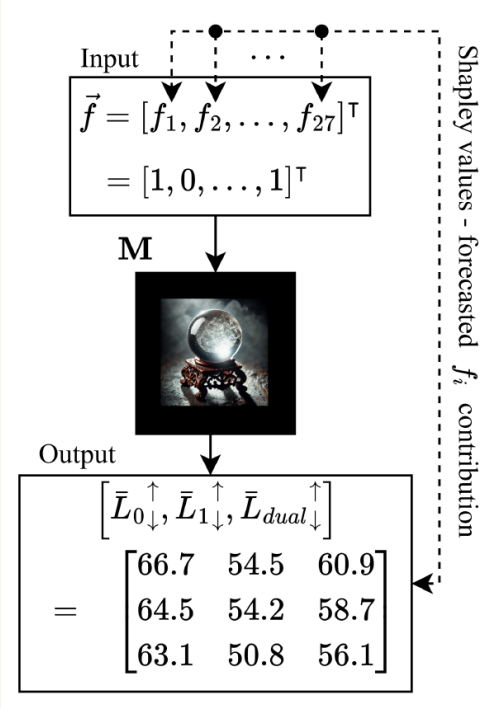


The SCENARIO

- Embedded electronic device from a partner company
- Dual CPU System with CPU load measurements for CPU 1, CPU 2 and Total (3 measurements summarum)
- 500 ms measurement interval, 27 tasks available for toggling
- Variable threshold of load acceptable on the device, depending on the configuration (can be different for both different configurations and cores)

The ALGORITHM

- Our tasks are represented by a binary vector of tasks, each task denoted 1 for ON, 0 for OFF
- A model, be it a neural network, GB, linear regression, fitted via supervised training, forecasts the predicted values of Core 1, 2 and total utilization
- We utilize the conformal prediction framework to generate lower and upper bounds for the three forecasts
- We execute SHAP analysis on the values



On conformal prediction - visualization

Imagine you're throwing a dart at a board blindfolded, but someone whispers, “Don’t worry, I’ll make sure the bullseye is within this big circle I drew!” That’s conformal prediction: it gives you a *region*, not a single point, where the bullseye (true value) *probably* is. It’s like wrapping your predictions in a safety net, saying, “Hey, 90% of the time, the answer’s in here!”

The primary clever point is balancing *accuracy* (the dart being in the circle) and *precision* (not making the circle *way too big*).

Standard conformal prediction draws a circle around the bullseye to ensure your dart (prediction) lands inside most of the time. Split conformal prediction refines this process by splitting your data: one part trains a model to estimate where the dart is likely to land, and the other part evaluates the model to adjust the circle size.

This split allows for sharper, data-driven confidence regions while still maintaining the guarantee that the true value lies within the region a specified percentage of the time. It’s a smarter, more efficient way to balance accuracy and precision.

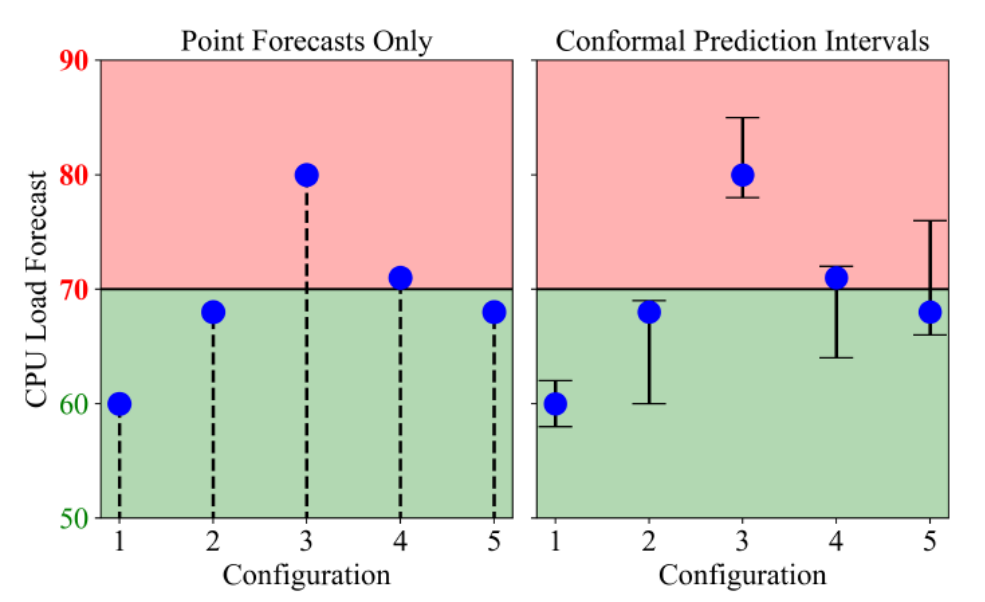
How Split CP works here

1. Denote the desired miscoverage rate, typically $\alpha = 0.1$
2. Split the total dataset into TRAIN, TEST, CALIBRATION
3. Train the base regressor on TRAIN
4. Train the lower quantile regressor $QR_{\alpha/2}$ and the upper regressor $QR_{\{1-\alpha/2\}}$
5. Compute residuals from CALIBRATION by finding maximal differences between y and the upper regressor and between y and the lower regressor*
6. Find the corresponding quantile $\hat{q}$ as the 1 - α** quantile of the residuals
7. Generate lower bounds as $QR_{\alpha/2}$- $\hat{q}$ and upper bounds as $QR_{1-\alpha/2} + \hat{q}$

* = other scoring metrics work too!

** = finite sample correction factor excluded here

Yielding...


Point Forecasts Only
Conformal Prediction Intervals
CPU Load Forecast
90
80
70
60
50
1
2
3
4
5
Configuration
Configuration

SHAP Analysis

- Shapley values are an approach to explaining outputs of machine learning models rooted in cooperative game theory.

- Each of the model inputs, that is, each one of the tasks (with binary value 0 or 1) gets its own Shapley value
- Core 0/1/dual
- prediction/upper/lower
- In our case, each inference run generates 27 (task) times 3 (cpu 0/1/2) times 3 (pred/lower/upper) = 243 values
- NP Hard problem, but great approximations exist!

For a set of features Φ and a prediction function γ, the Shapley value for feature $i \in \Phi$ is calculated as the weighted average of the increase of the output value when i is active with the group of features $S \subseteq \Phi$ versus when i is inactive with group S. This averaging is done over all possible subsets S where i is not in, and formally the definition is:

$$\overbrace{\phi_i}^{\text{i's Shapley}} = \sum_{S \subseteq \Phi \setminus \{i\}} \underbrace{\frac{|S|!(|\Phi| - |S| - 1)!}{|\Phi|!}}_{\text{S's weight}} \overbrace{\Big[\gamma(S \cup \{i\}) - \gamma(S)\Big]}^{\text{i's marginal contribution}}, \tag{4}$$

Testing results

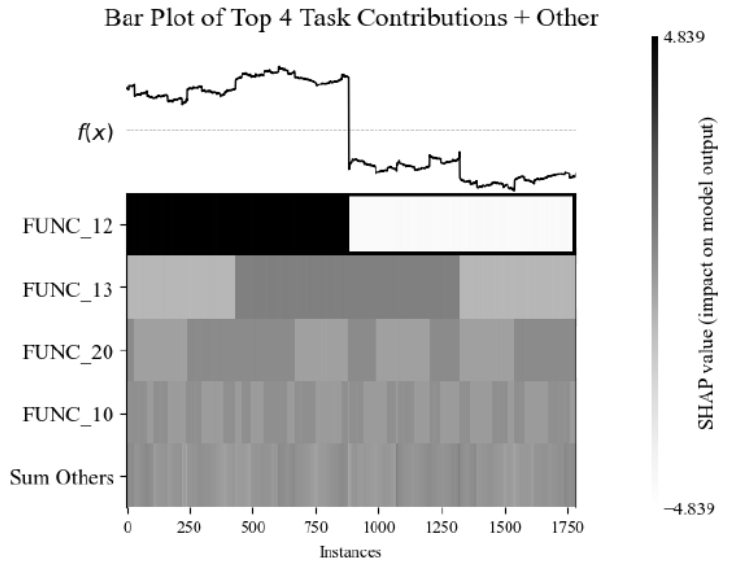


Fig. 10. Shapley value summary for predictions on Core 0, un-ablated baseline test dataset, 1784 testing samples.

- We analyzed which of the 27 tasks contributed to the majority of the load and discovered 4 which overwhelm
- We tested coverage rate, interval length and prediction error for the conformal prediction component
- We tested additivity, sensitivity and separate ablation for the Shapley value component

TABLE II

COVERAGE, MEAN INTERVAL LENGTH AND PREDICTION ERROR FOR THE MODEL M, C0 - CORE 0, C1 - CORE 1, DUAL - DUAL CORE

	Coverages			Mean Interval Lengths			MSE		
Ablated Function	C0	C1	Dual	C0	C1	Dual	C0	C1	Dual
1	0.89	0.89	0.89	1.32	7.94	4.66	0.05	0.47	0.17
2	0.89	0.89	0.89	1.32	7.93	4.66	0.04	0.45	0.16
3	0.89	0.89	0.89	1.27	7.74	4.58	0.04	0.46	0.16
4	0.89	0.89	0.89	1.32	7.92	4.65	0.04	0.45	0.16
5	0.89	0.89	0.90	1.32	7.93	4.66	0.06	0.48	0.17
6	0.89	0.88	0.89	1.32	7.93	4.66	0.05	0.44	0.16
7	0.89	0.89	0.89	1.27	7.94	4.67	0.06	0.47	0.17
8	0.89	0.89	0.89	1.32	7.93	4.65	0.05	0.46	0.17
9	0.89	0.88	0.89	1.32	7.93	4.67	0.04	0.47	0.17
10	0.89	0.89	0.89	1.33	7.98	4.69	0.05	0.53	0.19
11	0.89	0.89	0.89	1.32	7.92	4.65	0.04	0.47	0.17
12	0.89	0.89	0.90	2.60	17.94	10.26	2.44	70.85	23.76
13	0.88	0.89	0.89	1.48	8.07	4.75	0.26	2.66	1.06
14	0.89	0.89	0.89	1.34	7.93	4.64	0.05	0.45	0.16
15	0.89	0.89	0.90	1.33	7.93	4.66	0.04	0.47	0.16
16	0.88	0.89	0.88	1.32	7.91	4.64	0.06	0.48	0.17
17	0.89	0.88	0.89	1.32	7.90	4.63	0.05	0.49	0.17
18	0.89	0.89	0.89	1.32	7.92	4.66	0.04	0.46	0.16
19	0.89	0.89	0.89	1.32	7.92	4.66	0.04	0.45	0.16
20	0.89	0.88	0.89	1.33	8.00	4.73	0.08	0.75	0.28
21	0.89	0.88	0.89	1.32	7.93	4.66	0.06	0.47	0.16
22	0.89	0.89	0.89	1.33	7.93	4.67	0.06	0.46	0.16
23	0.89	0.88	0.89	1.34	7.93	4.66	0.07	0.48	0.18
24	0.89	0.89	0.89	1.33	7.93	4.66	0.05	0.47	0.16
25	0.89	0.89	0.89	1.35	7.99	4.71	0.06	0.47	0.16
26	0.88	0.88	0.89	1.35	8.01	4.73	0.06	0.46	0.16
27	0.89	0.88	0.89	1.32	7.93	4.67	0.04	0.46	0.16
None	0.89	0.89	0.89	1.32	7.92	4.66	0.06	0.46	0.16

TABLE III

MEAN ABSOLUTE ADDITIVITY ERROR TABLE FOR SHAPLEY VALUES ACROSS ABLATIONS AND BASELINE $\left|\overline{\text{Base}_{est}} - \text{Base}_{est}\right|$

	Core 0			Core 1			Dual Core		
Ablated Function	Lower	Upper	Pred.	Lower	Upper	Pred.	Lower	Upper	Pred.
None	0	0	0.17	0	0	0.17	0	0	0.14
1	0	0	0.12	0	0	0.16	0	0	0.13
2	0	0	0.16	0	0	0.16	0	0	0.13
3	0	0	0.12	0	0	0.15	0	0	0.13
4	0	0	0.19	0	0	0.19	0	0	0.14
5	0	0	0.15	0	0	0.19	0	0	0.14
6	0	0	0.13	0	0	0.18	0	0	0.14
7	0	0	0.19	0	0	0.19	0	0	0.14
8	0	0	0.13	0	0	0.18	0	0	0.14
9	0	0	0.16	0	0	0.18	0	0	0.14
10	0	0	0.19	0	0	0.22	0	0	0.17
11	0	0	0.16	0	0	0.2	0	0	0.16
12	0	0	0.62	0	0	1.1	0	0	0.86
13	0	0	0.24	0	0	0.33	0	0	0.27
14	0	0	0.13	0	0	0.16	0	0	0.14
15	0	0	0.18	0	0	0.17	0	0	0.14
16	0	0	0.12	0	0	0.18	0	0	0.13
17	0	0	0.18	0	0	0.19	0	0	0.16
18	0	0	0.15	0	0	0.17	0	0	0.13
19	0	0	0.18	0	0	0.18	0	0	0.14
20	0	0	0.18	0	0	0.21	0	0	0.17
21	0	0	0.16	0	0	0.16	0	0	0.13
22	0	0	0.16	0	0	0.2	0	0	0.15
23	0	0	0.12	0	0	0.16	0	0	0.13
24	0	0	0.15	0	0	0.14	0	0	0.11
25	0	0	0.13	0	0	0.16	0	0	0.12
26	0	0	0.14	0	0	0.2	0	0	0.15
27	0	0	0.16	0	0	0.18	0	0	0.13

Conclusions and Future Work

- Developed an integrated CPU load forecasting framework combining conformal prediction and Shapley value-based interpretability
- Achieved robust uncertainty quantification and actionable task-level insights through Shapley value analysis
- Demonstrated consistent prediction accuracy across configurations, with key functions retaining importance under ablation
- Plan to extend the model library with alternative conformal predictors (GBM, RF, QRF) to improve robustness in diverse settings
- Future datasets will include richer system-level metrics (e.g., interrupts, I/O, memory, context switching) for holistic modeling
- Will explore incremental learning (e.g., elastic weight consolidation) to retain knowledge during continual updates
- Intend to build a real-time application interface supporting intuitive task management and embedded device integration
- Will investigate training-conditional coverage techniques to enhance uncertainty calibration at the per-sample level