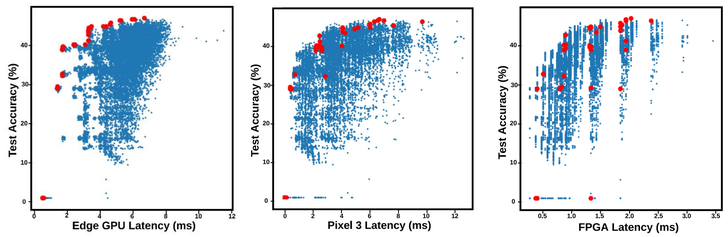
# Hardware/Software Co-design for embedded AI with AutoML

Nina Bretz, Thomas Rocznik*, Dennis Rieber, Christoph Schorn, Thomas Elsken
Bosch Center for Artificial Intelligence, Germany *Robert Bosch LLC, USA

## Motivation

- Bringing **AI algorithms to the edge** requires executing neural networks (NN) on resource-constrained embedded hardware (HW)
- How should the NN architecture look like to fulfil the application requirements while making optimal use of the available compute resources?
- The **target HW is often not fixed** but can be chosen from a list of available devices or can even be parameterized as well
- Optimal choice of NN and HW depend on each other, i.e., joint optimization is required (see Figure 1)
- We propose to extend AutoML systems to **jointly optimize the hardware configuration and NN architecture**, thus addressing the **hardware/software co-design problem in an automated fashion**.



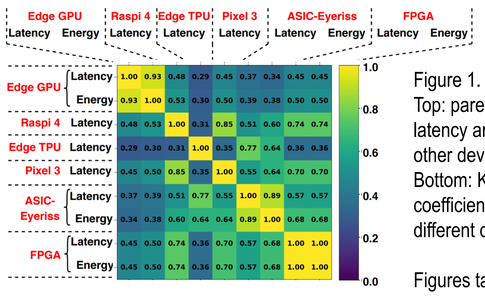




Figure 1.
Top: pareto-optimal NN w.r.t. GPU latency are not necessarily optimal on other devices.
Bottom: Kendall rank correlation coefficient between HW metrics on different devices.

Figures taken from [1].

## Background

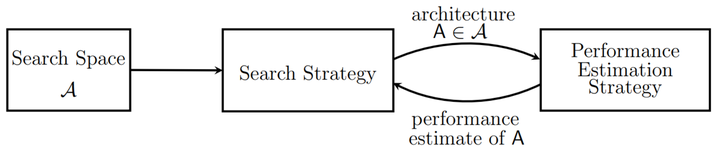



Figure 2. Overview of the different components of NAS algorithms. Figure from [2].

- **Neural architecture search (NAS)** [2] aims at automatically discovering optimal neural network architectures for a given task
- NAS algorithms typically consist of the following components (see Figure 2)
- Evolutionary algorithms (Figure 3) are a commonly used approach for NAS [3,4] due to their simplicity, flexibility & parallelizability. Can also be naturally extended to multi-objective optimization [4,5,6].
- Hardware/Software Co-design benefits from an automated toolchain because [7]:
  - Overutilization of compute resources prevents the model from running efficiently on the given HW, while underutilization can mean that the inference capabilities of the AI algorithm are unnecessarily curtailed
  - Manual choice of NN building blocks and their parametrization is not straightforward, due to the huge size of the design space
  - Choice between different HW targets further increases the system design space

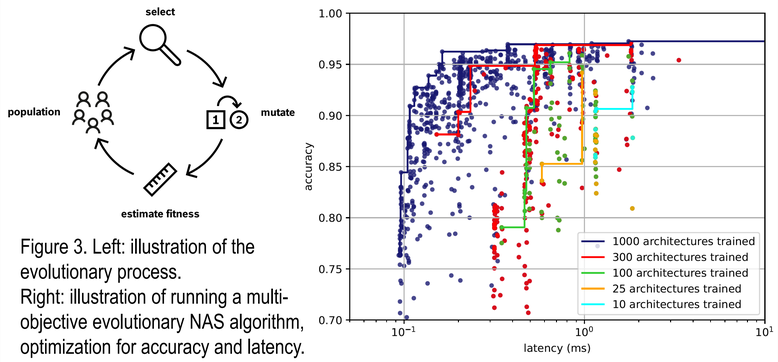



Figure 3. Left: illustration of the evolutionary process.
Right: illustration of running a multi-objective evolutionary NAS algorithm, optimization for accuracy and latency.

## Method

- We extend the simple evolutionary NAS method [3] to multi-objective optimization by using non-dominated sorting [6] for ranking candidates
- Beside optimizing the NN architecture, we also optimize the HW configuration, i.e., we **jointly optimize the NN and target HW configuration**
- We consider in this study a **design-time parametrizable hardware accelerator:** HW configurations which are considered are low-level hardware design choices such as **buffer memory sizes or the number of processing elements** (PE for the multiply-accumulated operations)
- Neural network architectures are deployed to a virtual model of the parametrizable accelerator to obtain fast predictions of the resulting hardware

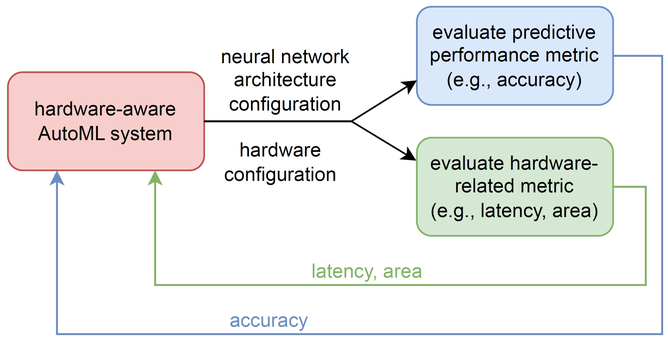



Figure 4. We employ a hardware-aware AutoML system that proposes candidate solutions for both the NN architecture and the HW configuration. This joint (NN architecture, hardware) configuration is then evaluated: we measure both metrics related to the predictive performance of the NN (e.g., by means of accuracy) as well as to the HW. HW-related metrics indicate how efficiently the chosen NN can be run on the selected HW configuration (e.g., by means of latency), as well as how costly it is to produce the HW (e.g., by means of silicon area). These metrics are then fed back into the AutoML system as objective functions which are being optimized.

## Experiments

- In this study we consider a **design-time parametrizable hardware accelerator** to perform a **classification problem based on radar sensor signals**.
- During evolution, we mutate 3 types of parameters: (i) NN architecture, (ii) hyperparameters for training (e.g., learning rate), (iii) the HW configuration

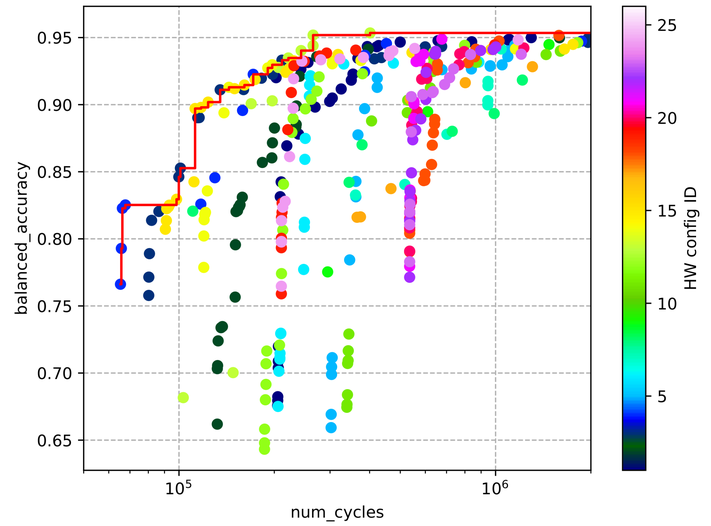



Figure 5. We fix the HW configuration and run NAS on each fixed HW configuration independently. The red curve shows the Pareto-optimal solution across 26 fixed HW configurations. One can see that different HW configurations contribute to the optimal solution, indicating that considering only a single fixed HW configuration is suboptimal.

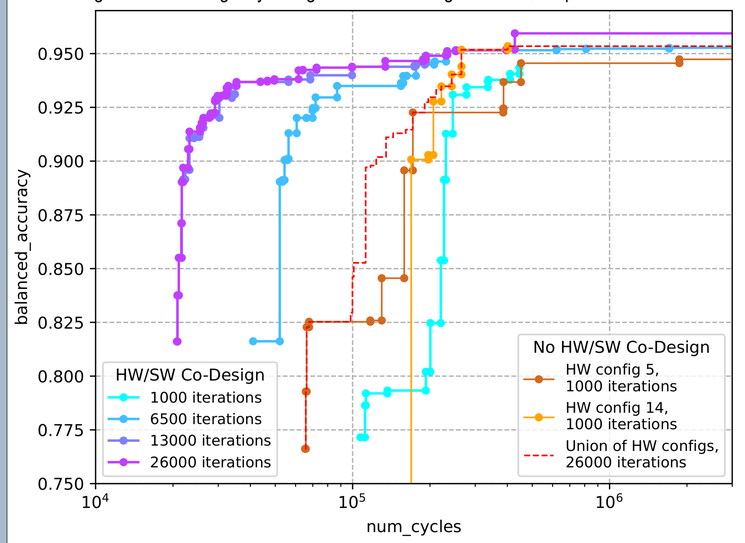



Figure 6. We jointly optimize NN and HW architectures and compare to running NAS independently for 26 HW configurations (red) and 2 cherry-picked fixed HW configurations (orange, brown). With only 25% of the budget (6,500 iterations), the joint optimization already clearly outperforms running NAS independently for multiple fixed HW configurations.

## References

[1] Li et al., HW-NAS-Bench: Hardware-Aware Neural Architecture Search Benchmark, ICLR 2021
[2] Elsken et al., Neural Architecture Search: A Survey , JMLR 2019
[3] Real et al., Regularized Evolution for Image Classifier Architecture Search, AAAI 2019
[4] Elsken et al., Efficient Multi-Objective Neural Architecture Search via Lamarckian Evolution, ICLR 2019
[5] Benmeziane et al.,, A Comprehensive Survey on Hardware-Aware Neural Architecture Search, IJCAI 2021
[6] Deb et al., A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II, IEEE Transactions on Evolutionary Computation 2002
[7] Bringmann et al., Automated HW/SW Co-Design for Edge AI: State, Challenges and Steps Ahead, CODES/ISSS 2021