Multi-Objective Bayesian Optimization of Deep Neural Networks for Deployment on Microcontrollers

Lehrstuhl für Informatik 12
(Hardware-Software-Co-Design)

Friedrich-Alexander-Universität Erlangen-Nürnberg
Mark Deutel, Frank Hannig, and Jürgen Teich

Motivation

TinyML – Machine Learning on the Edge

Efficiency

- Processing of data close to the sensor
- Re-usage of (hardware) resources required to drive the sensor

Reliability

- No communication via error prone network required
- Short, predictable “round-trip time"

TinyML

Cost

- Exploitation of already available cheap consumer-grade hardware
- Low energy footprint

Privacy

- Possibly confidential data is processed on the sensor node
- No connection to external cloud or server required

Motivation

DNN Deployment on Microcontrollers – An easy task?

Deep Neural Network Architectures[1]

Metrics	AlexNet	VGG 16	ResNet 50
# Layers	8	16	54
Total Weights	61 M	138 M	25.5 M
Total MACs*	724 M	15.5 G	3.9 G

1. Sze, Vivienne, Yu-Hsin Chen, Tien-Ju Yang, and Joel Emer. „Efficient Processing of Deep Neural Networks: A Tutorial and Survey". 2017.

Target Micro Controllers

Metrics	Raspberry Pi Pico	Arduino Nano 33 BLE Sense
Processor	ARM Cortex M0+	ARM Cortex M4
Clock Speed	133 MHz	64 MHz
Flash memory	2 MB	1 MB
SRAM	256 KB	256 KB

Significant gap between DNN requirements and available resources

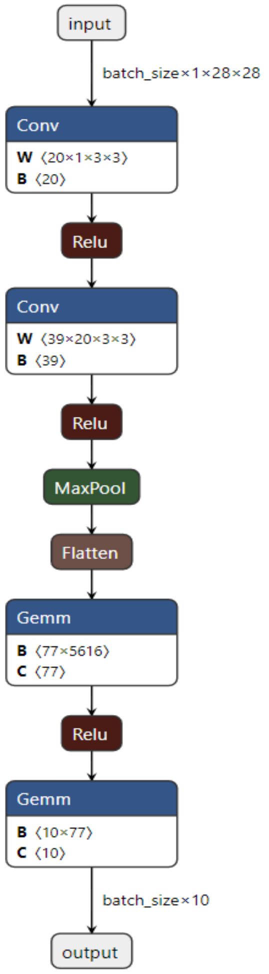
- Low processor speed vs. large number of mathematical operations
- Strict memory limitations vs. large number of weights and big inputs/feature maps
- High precision floating point datatypes vs. hardware often focused on integer arithmetic

* Multiply-Accumulate Operations

Motivation

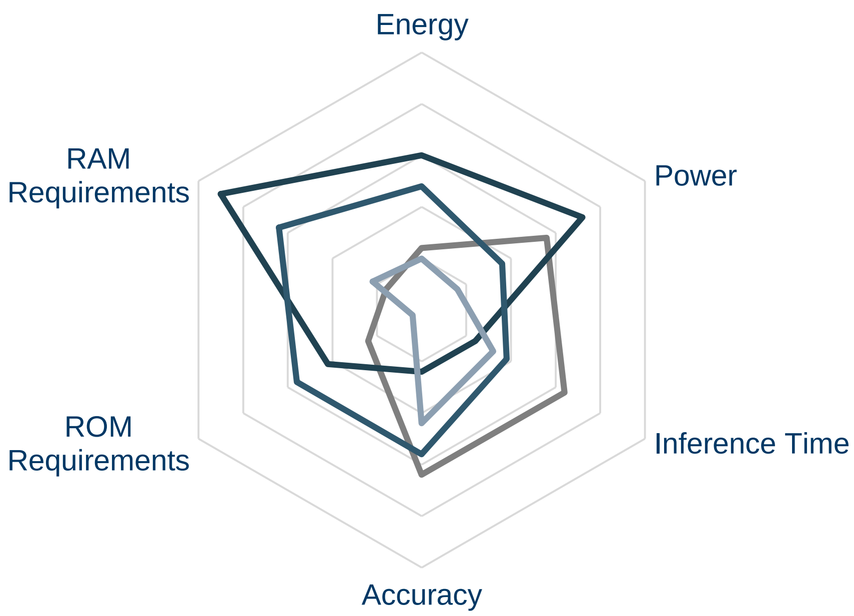
DNN Deployment on Microcontrollers – An Optimization Problem

DNN Architecture, Dataset



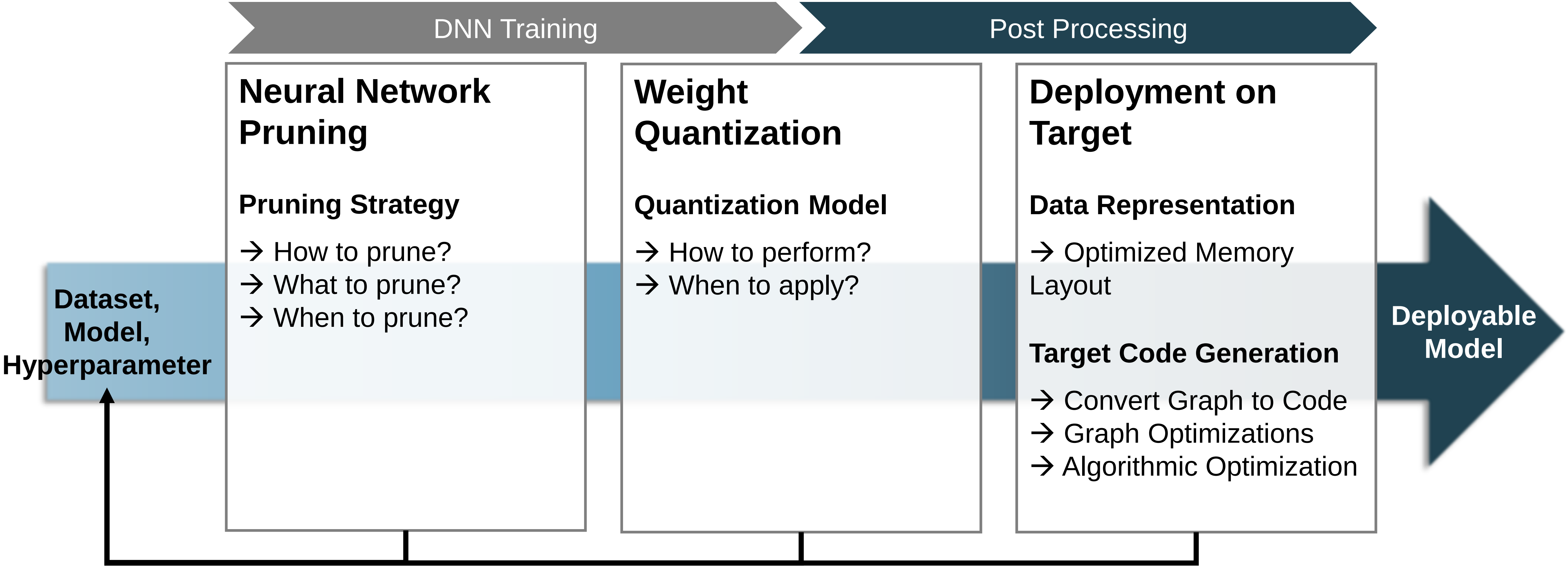
DNN Compression and Deployment

Deployment on Target System



Deployment of Deep Neural Networks on Microcontrollers

A Fully-Automated End-to-End DSE Pipeline for DNN Deployment[1]



Suggest next set of Hyperparameters (parameter space) using Multi-Objective Optimization based on performance metrics (objective space: accuracy, ROM, RAM, FLOPS)

1. Deutel, Mark, et al. "Deployment of Energy-Efficient Deep Learning Models on Cortex-M based Microcontrollers using Deep Compression." *arXiv preprint arXiv:2205.10369* (2022).

Agenda

- Deployment of DNNs on Microcontroller Targets
  - Network Pruning
  - Weight Quantization
  - Microcontroller Deployment
- Optimizing DNNs using Multi-Objective Optimization
  - Introduction to Multi-Objective Optimization
  - Multi-Objective Bayesian Optimization
  - Evaluation of Use-Cases
- Conclusion

Deployment of DNNs on Microcontroller Targets

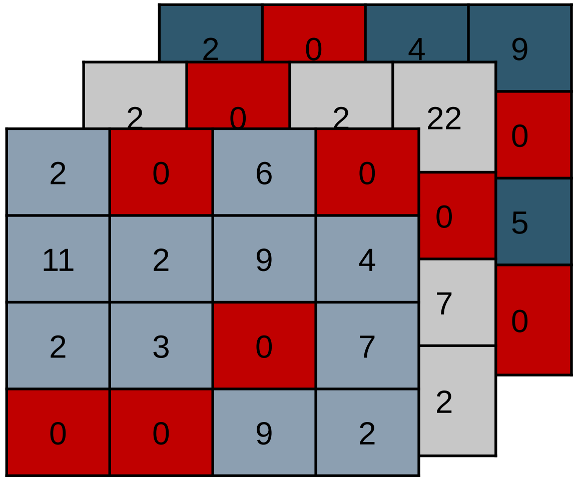
Network Pruning

Pruning Strategy

Understanding: Neural Networks are extremely over-parametrized and have a lot of redundancies in their parameters

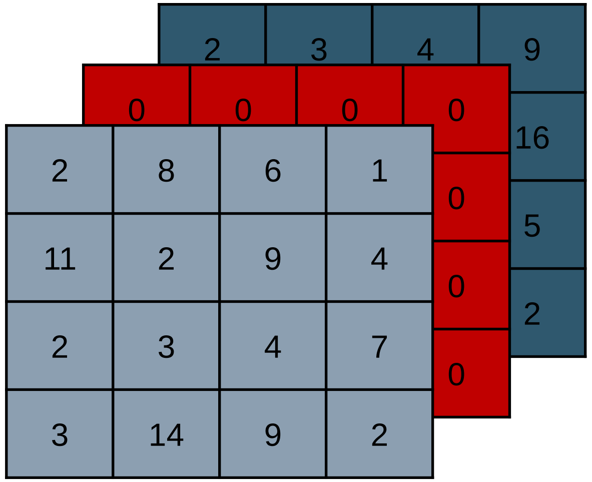
Element-Wise (Unstructured) Pruning[1]

- Set single weights to zero
- Sparse data structures remain at the end of training



Structured Pruning[2]

- Set whole structures of weights to zero
- Structures (and their dependencies) can be removed at the end of training



1. LeCun, Yann, John S Denker, and Sara A Solla. “Optimal Brain Damage“, 1989.
2. Anwar, Sajid, Kyuyeon Hwang, and Wonyong Sung. “Structured Pruning of Deep Convolutional Neural Networks“. 2015.

Network Pruning

Pruning Heuristics

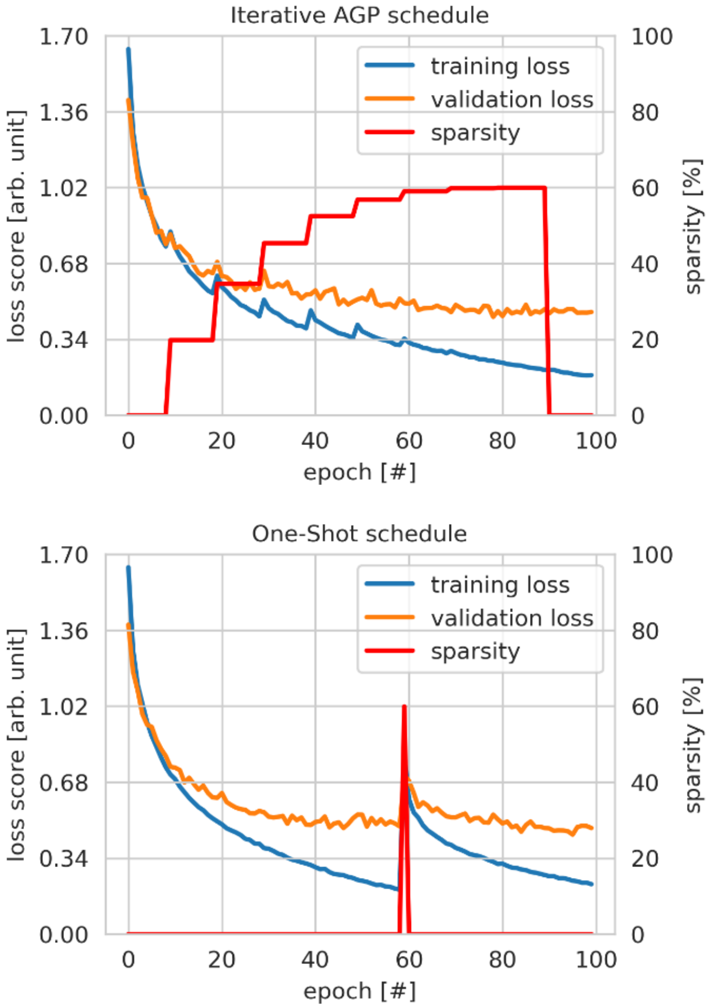
- Are used as an approximation to decide which structures/elements to remove
  - Magnitude/L-Norm-based[1],
  - Gradient-based[2],
  - Average percentage of Zeros[3] (ApoZ) in activation tensors
  - Attribution-based heuristics[4] (Explainable AI)

1. Han, Song, Jeff Pool, John Tran, und William J. Dally. “Learning both Weights and Connections for Efficient Neural Networks“. 2015.
2. Molchanov, Pavlo, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. “Pruning Convolutional Neural Networks for Resource Efficient Inference“. 2017.
3. Hu, Hengyuan, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. “Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures“. 2016.
4. Sabih, Muhammad, Frank Hannig, and Juergen Teich. “Utilizing explainable AI for quantization and pruning of deep neural networks.“ 2020.

Network Pruning

Pruning Schedule

- Defines **when** and **how often** pruning is applied during training
- **Iterative Pruning:**
  - Prune multiple times during training
  - Increase sparsity starting with a low value
  - Automated Gradual Pruning[1] (AGP) algorithm
- **One-Shot Pruning:**
  - Prune one time at the end of training
  - Enforce all sparsity at once



1. Zhu, Michael, and Suyog Gupta. “To prune, or not to prune: exploring the efficacy of pruning for model compression“. 2017.

Weight Quantization

Quantization Schema

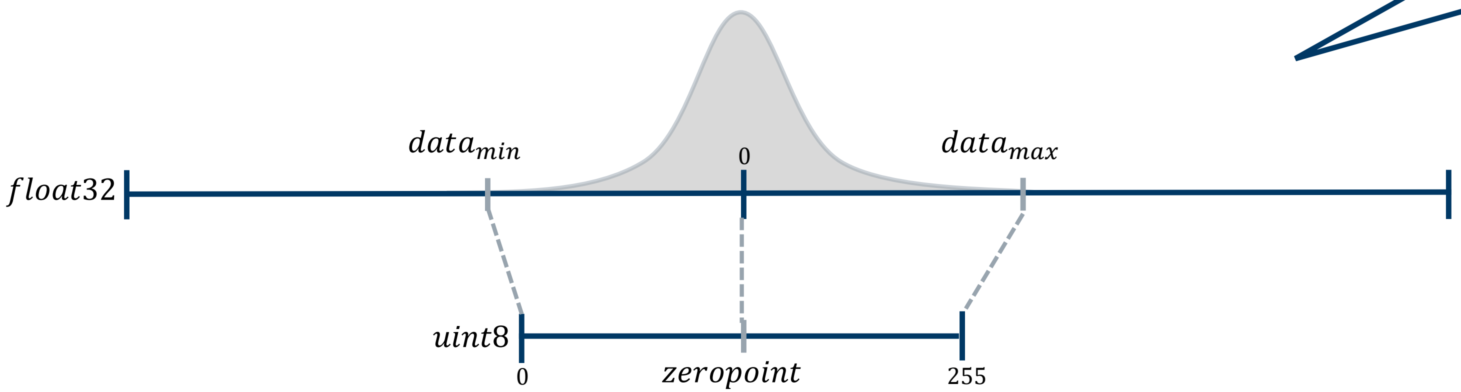
Instead of using high-precision floating-point arithmetic to store trained weights, map them to integer space instead

- Affine, linear mapping from **32-bit floating point** space to **8-bit unsigned integer** space using (trainable) **scale** and **zero point** parameters[1]
- Both weight and activations tensors can be quantized (i.e. partial and full quantization)

$$val_{uint8} = \mathrm{clamp}\left(\left\lfloor \frac{val_{fp32}}{scale} \right\rceil + zeropoint\right)$$

$$scale = \frac{data_{max} - data_{min}}{255}, 0 \leq zeropoint \leq 255$$

Standardization of activation tensors is a great idea



1. see https://onnxruntime.ai/docs/performance/quantization.html

Weight Quantization

Quantization Strategy

Post Training Static Quantization[1]

- Perform Quantization once training has finished
- Use evaluation dataset from training to approximate zero points and scales

Quantization Aware Training[2]

- Fake quantized trainable weights and activations during training
- Use quantized parameters during forward passes to emulate quantization

1. Krishnamoorthi, Raghuraman. „Quantizing deep convolutional networks for efficient inference: A whitepaper". 2018.
2. Jacob, Benoit, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. „Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference". 2017.

Microcontroller Deployment

Overview

Conversion to Intermediate Representation

- **Convert graph-based** (ONNX) model to **target-platform-based** representation
  - **Weight** tensors → **Data** segment (Flash utilization)
  - **Network** structure → “Function” **Code** segment (Flash utilization)
  - **Activation** tensors → **Heap** (SRAM consumption)

Memory Allocation

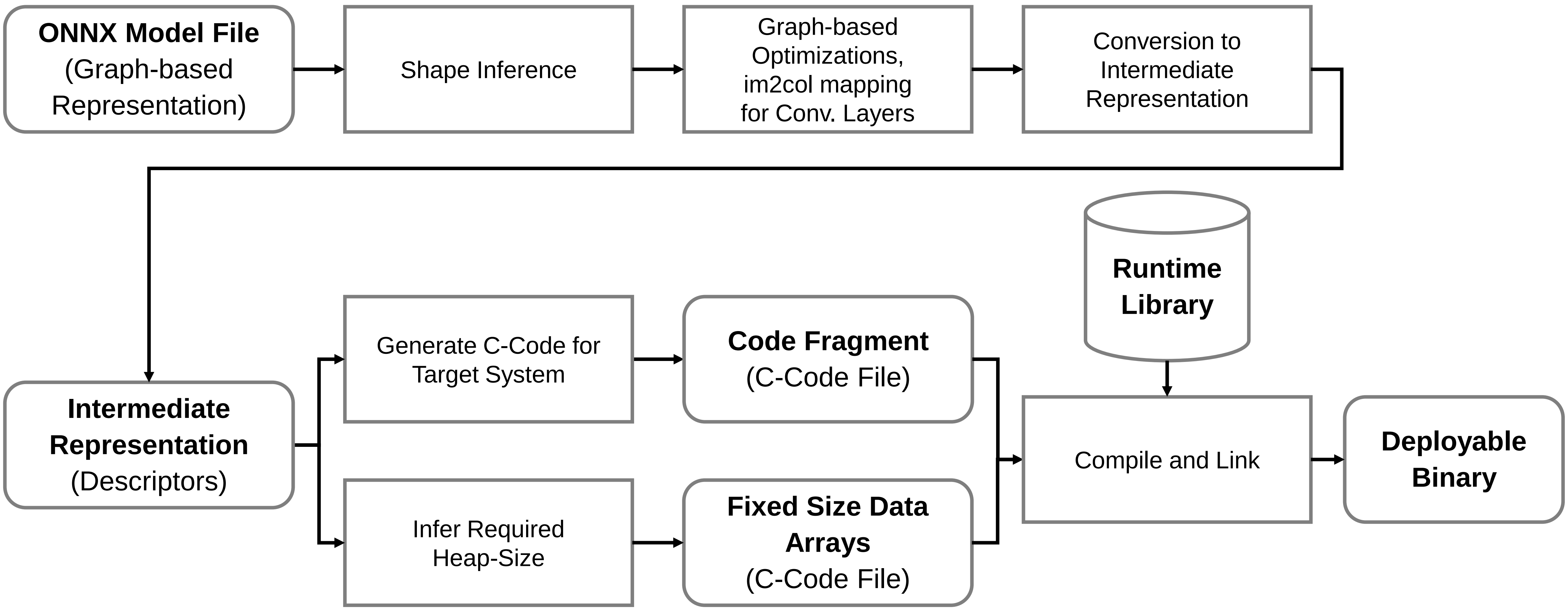
- **Allocate byte arrays** for both **weight and activation tensors**
- Dynamic memory allocation for activation tensors can be avoided by simulating offline
  → No malloc required at runtime

Code Generation + Compilation

- **Emit compilable C-code** representing the DNN (“ahead-of-time”)
- **Compile and link** with runtime library (optimized for each target platform)
- Implements common DNN layers based on the ONNX specification

Microcontroller Deployment

Deployment Pipeline



Optimizing DNNs using Multi-Objective Bayesian Optimization

Multi-Objective Optimization

Introduction (1)

- Maximize or minimize some objectives given a set of objective functions that map from parameter to objective space

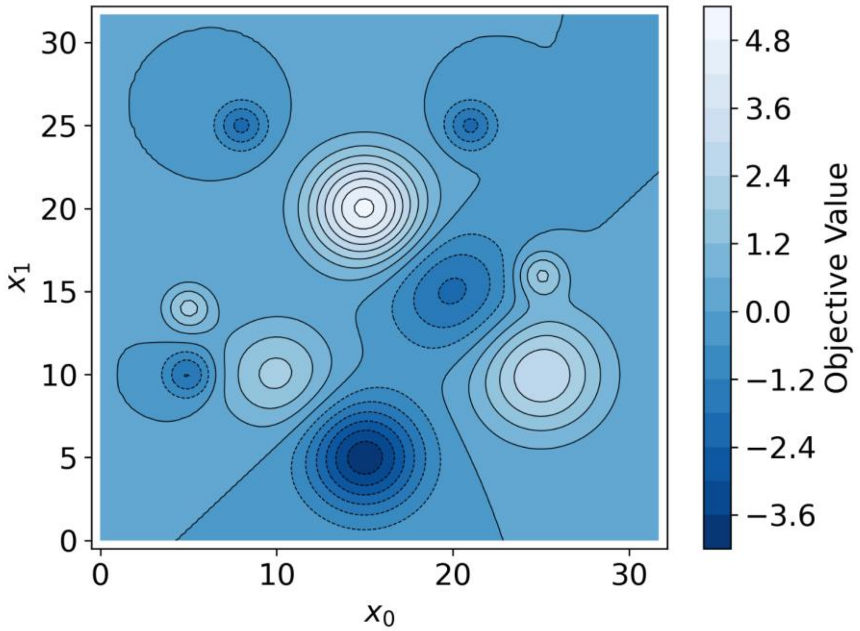
$$f(x) = [f_1(x), \dots, f_n(x)] \in \mathbb{R}^n, n \geq 2, x \in \mathbb{R}^d$$

  - There can be an additional set of constraints in the form of

$$g(x) \geq 0 \in \mathbb{R}^v$$

  - Usually, there exists no single solution x^* that maximizes/minimizes all objectives while also satisfying all constraints

- If a sample A compared to another sample B is equal in all objectives and better in at least one, A Pareto dominates B.
  - If a sample is not Pareto-dominated, it is Pareto optimal
  - If a sample meets all constraints, it is feasible
  - A set of feasible Pareto-optimal samples is called the feasible Pareto front

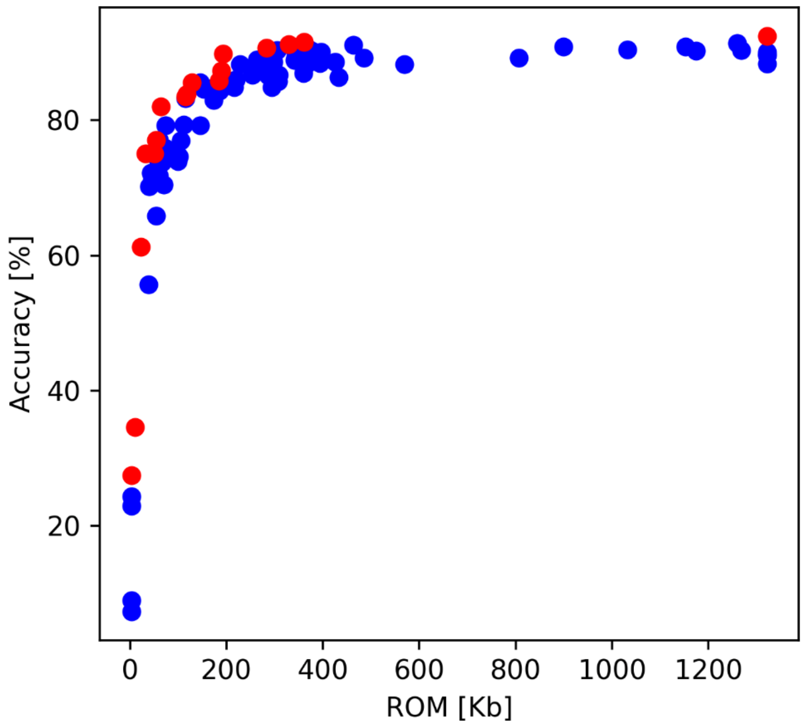


Passino, K. M. Biomimicry for optimization, control, and automation. Springer Science & Business Media, 2005.

Multi-Objective Optimization

Introduction (2)

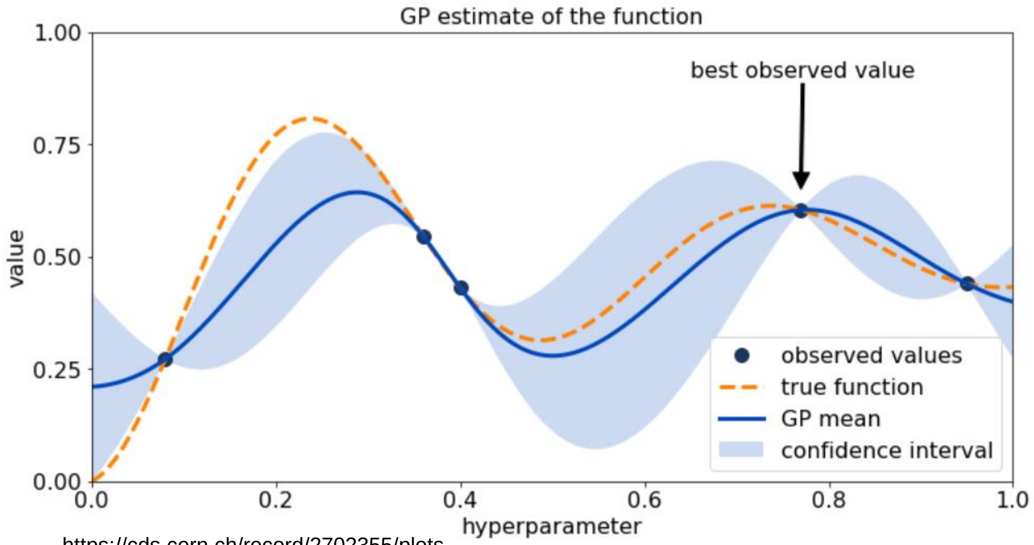
- The Hypervolume indicator is a measure of the quality of a (feasible) Pareto front and is calculated relative to a reference point R
  - The Hypervolume $h \in \mathbb{R}$ is in $[0,1]$, where a higher Hypervolume value indicates a better coverage of the target space
  - To compare the Hypervolume of two Pareto-Fronts they have to …
    a. … be in the same target space
    b. … have the same reference point R



Multi-Objective Optimization

Bayesian Optimization[1]

- Improve decision making, i.e., suggesting the next parametrization x, by optimizing a surrogate model
  - Fit surrogate (Gaussian process) using previous samples (prior)
  - Solve optimization problem on surrogate model using an acquisition function
  - Suggest next parametrization to be evaluated
- Evaluate posterior of surrogate either analytical or by using Monte-Carlo (MC) sampling[2]
  - Evaluation of the acquisition function requires calculating an integral over the posterior distribution



https://cds.cern.ch/record/2702355/plots

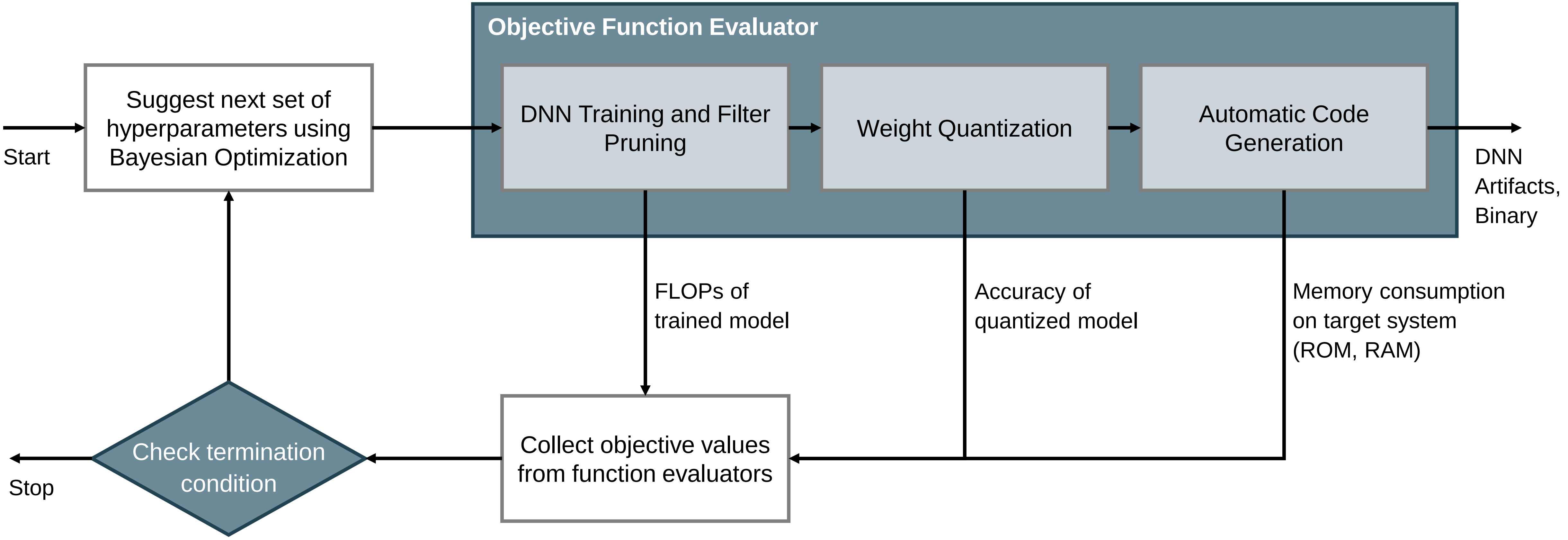
Expected Improvement[1]

$$\mathrm{EI}(X) = \mathbb{E}\max(f(x) - f^*, 0)$$

$$\alpha(x) \approx \frac{1}{N}\sum_{i=1}^{N} \max(\mathcal{E}_i - f^*, 0), \mathcal{E}_i \sim \mathbb{P}(f(x)|D)$$

1. Mŏckus, J. On Bayesian methods for seeking the extremum. In Optimization Techniques IFIP Technical Conference, pp. 400–404, 1975.
2. Wilson, J. T., Moriconi, R., Hutter, F., and Deisenroth, M. P. The reparameterization trick for acquisition functions. NIPS 2017 Workshop on Bayesian Optimization (BayesOpt 2017), 2017.

Optimizing DNNs using Multi-Objective Bayesian Optimization

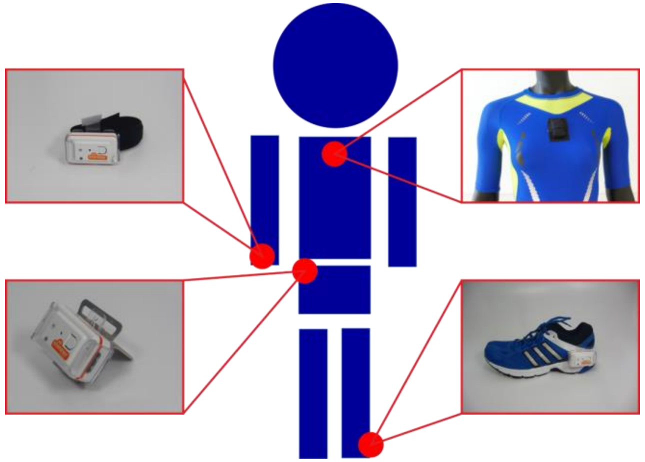


Evaluation

Datasets and Models

- CIFAR10[1] (32x32 RGB image classification) trained with scaled ResNet18[2]
  - 1.6M initial parameters
  - 3 residual networks (instead of 4)
- DaLiAc[3] (daily human activity, time series classification) trained with scaled MobileNetV3[4]
  - 2.3M initial parameters
  - Constant window length of size 1024, one class per window
  - Four sensor nodes with triaxial accelerometers and gyroscopes

https://www.cs.toronto.edu/~kriz/cifar.html

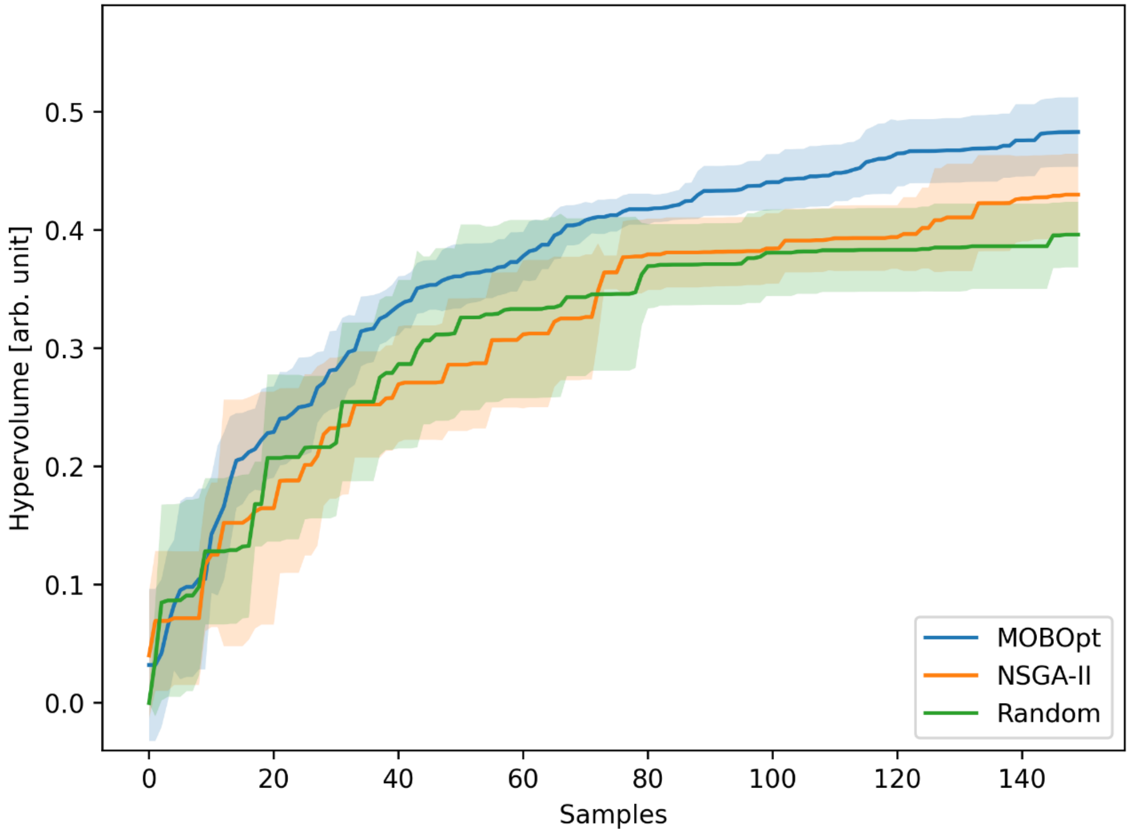


https://www.mad.tf.fau.de/research/activitynet/daliac-daily-life-activities/

1. Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. 2009
2. He, K., Zhang, X., Ren, S., and Sun, J. “Deep residual learning for image recognition”. 2016.
3. Leutheuser, H., Schuldhaus, and D., Eskofier, B. M. “Hierarchical, multi-sensor based classification of daily life activities: comparison with state-of-the-art algorithms using a benchmark dataset”. 2013.
4. Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., and Adam, H. “Searching for mobilenetv3”. 2019

Evaluation

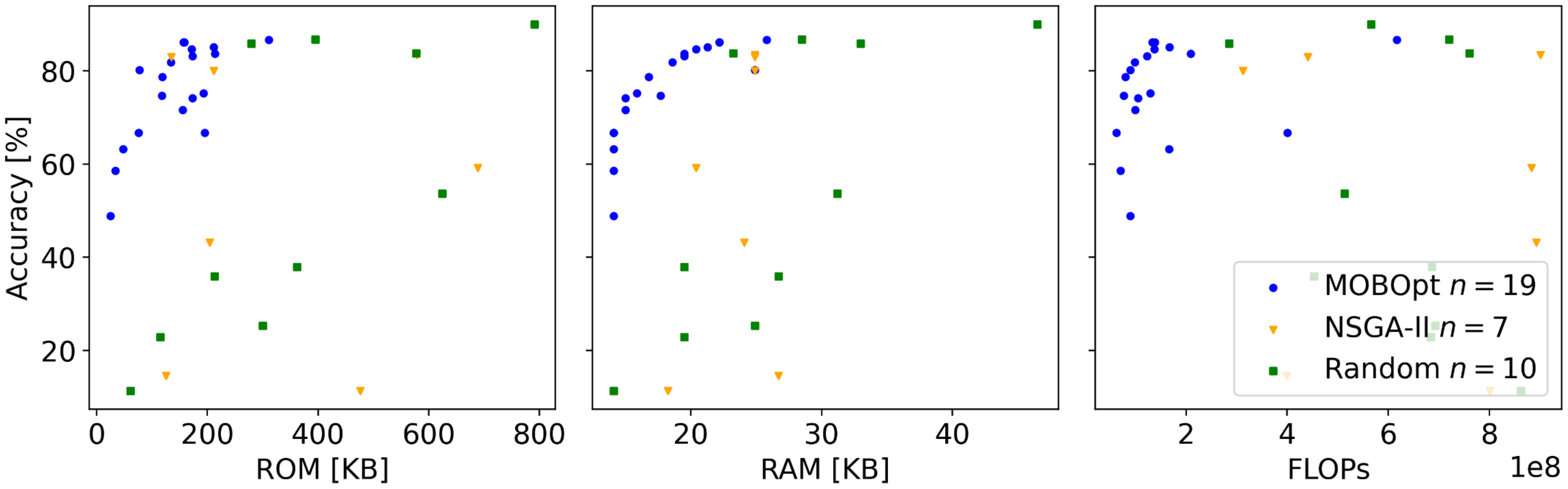
ResNet18, CIFAR10 (1)



Hypervolume Improvement for ResNet18, CIFAR10,
150 samples, 5 seeds each

Evaluation

ResNet18, CIFAR10 (2)

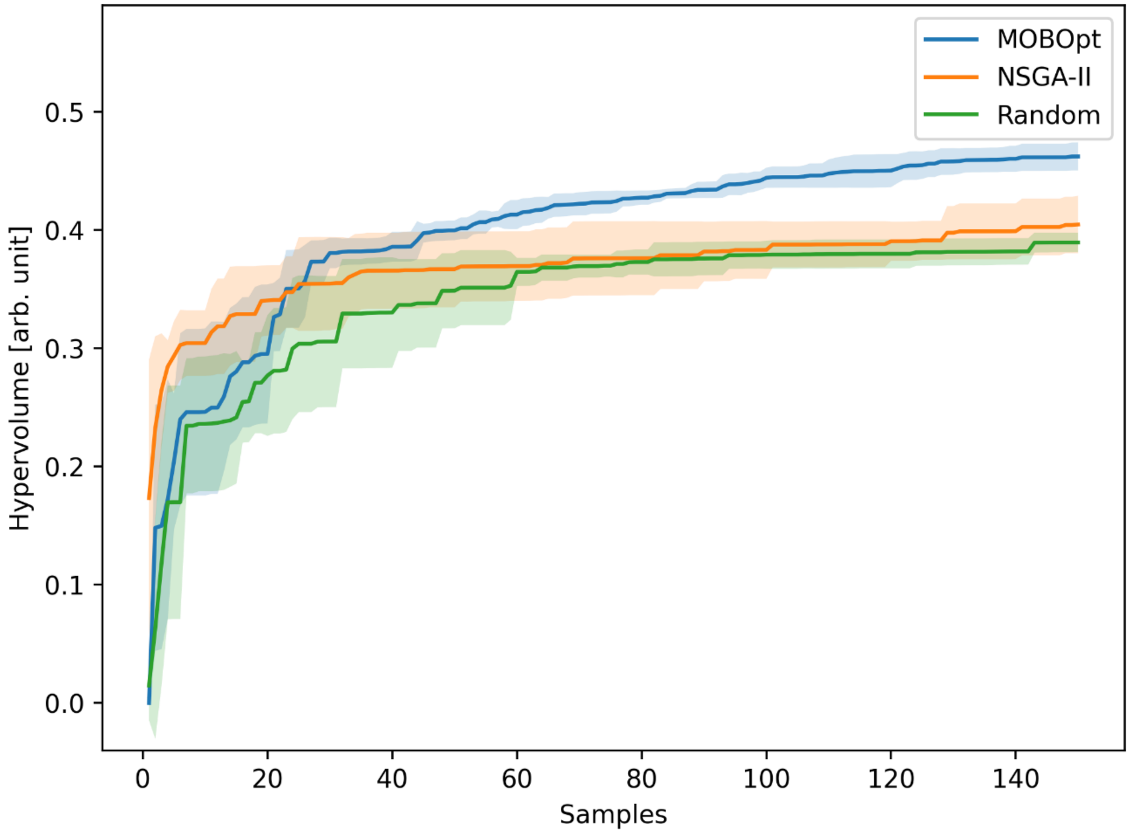


Feasible Pareto fronts for Bayesian and evolutionary solvers (exemplarily)

Constraints: < 1000 KB ROM, < 256 KB RAM, $< 1e^9$ FLOPs

Evaluation

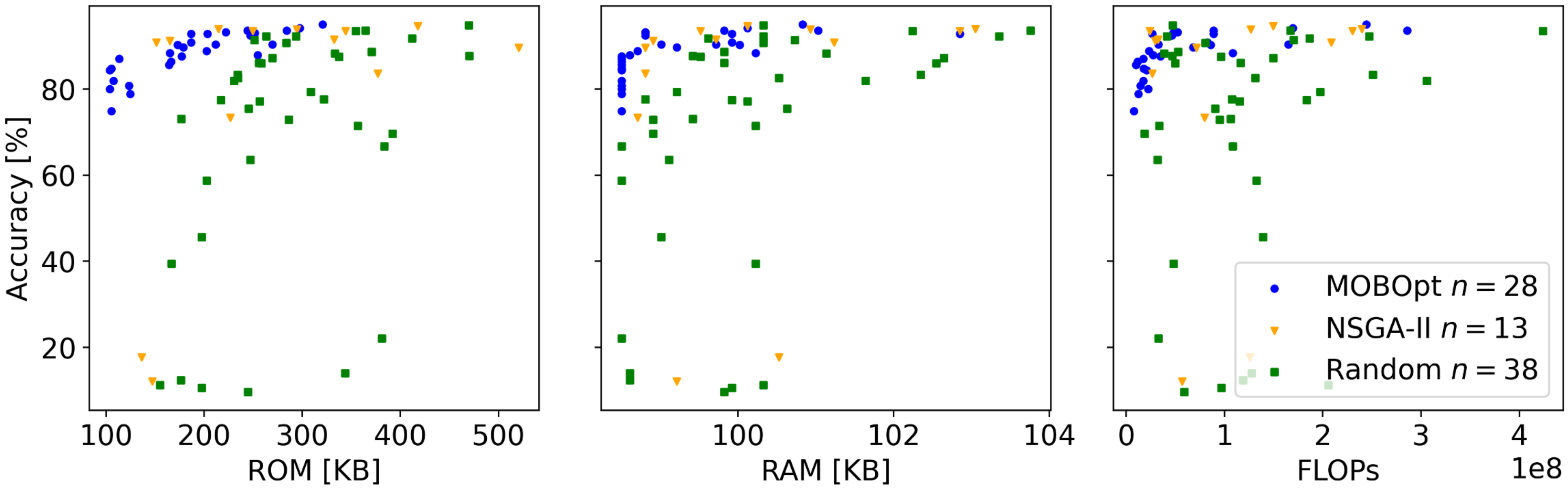
MobileNetV3, DaLiAc (1)



Hypervolume Improvement for MobileNetV3, DaLiAc,
150 samples, 5 seeds each

Evaluation

MobileNetV3, DaLiAc (2)



Feasible Pareto fronts for Bayesian and evolutionary solvers (exemplarily)

Constraints: < 1000 KB ROM, < 256 KB RAM, $< 1e^9$ FLOPs

Conclusion

Conclusion

- End-to-end DNN training, compression and deployment pipeline for microcontroller targets
  - Network pruning, Weight quantization, automated code generation
- Automated DSE for DNN hyperparameters using multi-objective optimization (AutoML)
  - Bayesian Optimization yields improved performance compared to evolutionary approaches in the given search budget
  - Optimization is limited by the high cost of the objective function evaluators (especially accuracy!)

Thank you for your attention!

Contact: mark.deutel@fau.de