technische universität dortmund

SFB 876 Providing Information by Resource-Constrained Data Analysis

Project A1

Data Mining for Ubiquitous System Software

Prof. Dr. Katharina Morik, Prof. Dr. Jian-Jia Chen

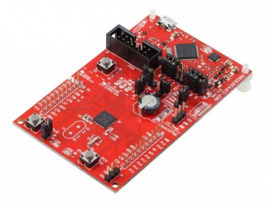
Problem

Phase 1 & 2

Central objectives

- Minimum prediction error
- Fast inference
- Small models
- Minimum resources assignment
- Minimum energy consumption
- Real-time guarantee

Limited to a single device



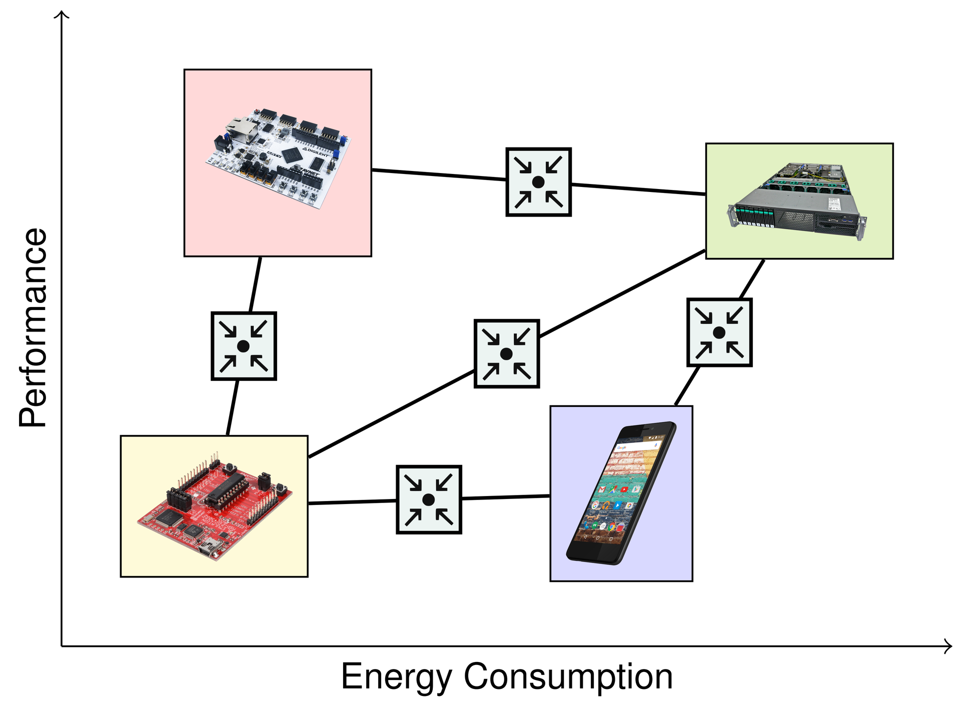
Phase 3

Extend analysis to

- Network of heterogenous devices
- Emerging new memory models
- Dynamic and adaptive execution of learning

Additional aspects

- Communication costs
- Synchronization costs
- Exploitation of heterogenous hardware
- Mapping models onto hardware



Planned Research

Hardware/Software Co-Design

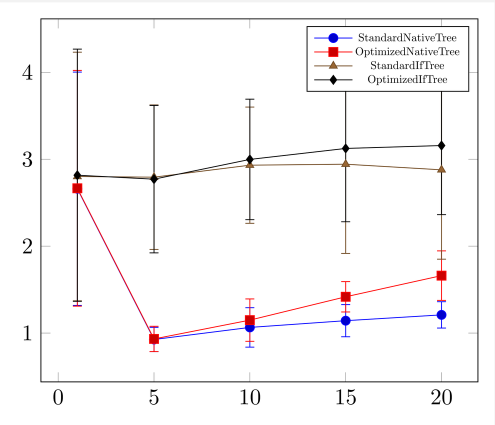
Goal Reduce the gap between hardware design and machine learning

- Analyse model application
- Analyse model learning
- Analyse hardware architecture
- Explore applications in A4 B2 B4 C3

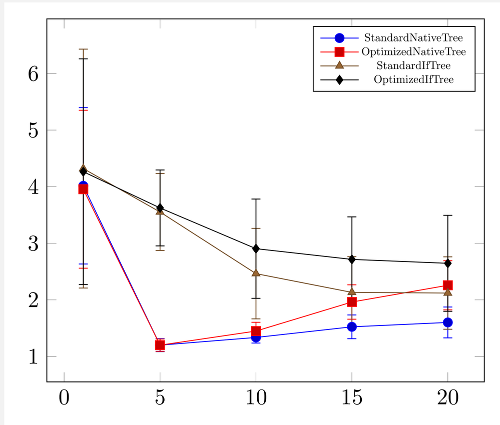
Preliminary Results [Buschjaeger; Chen; Chen; Morik; ICDM; 2018]

Architecture-specific implementation accelerates Random Forest application exploiting

- Model-dependent execution graph
- Data-dependent code synthesis
- Architecture-specific caching behaviour



Speed-up on Intel over tree size.



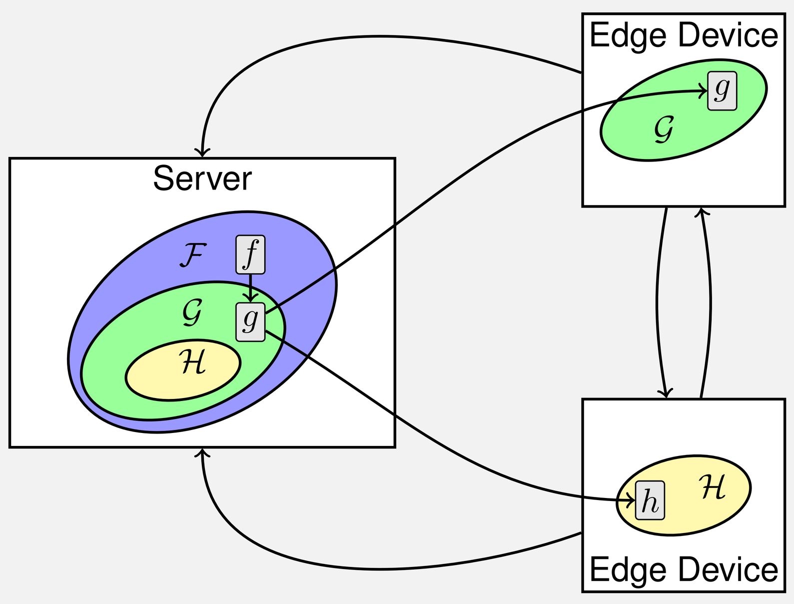
Speed-up on ARM over tree size.

Open questions

- Can we optimise compilation?
- Can we generalise to model learning?
- Can we include different models?
- Can we target FPGAs, GPUs, etc?

Distributed Machine Learning

Goal How to learn utilizing the edge?



Open questions

- How to post-process or prune f?
- Regularisation instead of post-processing?
- What are the statistical guarantees? B2
- What are the real-time guarantees? C3

Approaches

- Constrained model families $\mathcal{H} \subset \mathcal{G} \subset \mathcal{F}$
- Model learning via constrained optimization

$$g = \arg\min_{f \in \mathcal{G}} L(f, \mathcal{D})$$

- Model application via regularisation

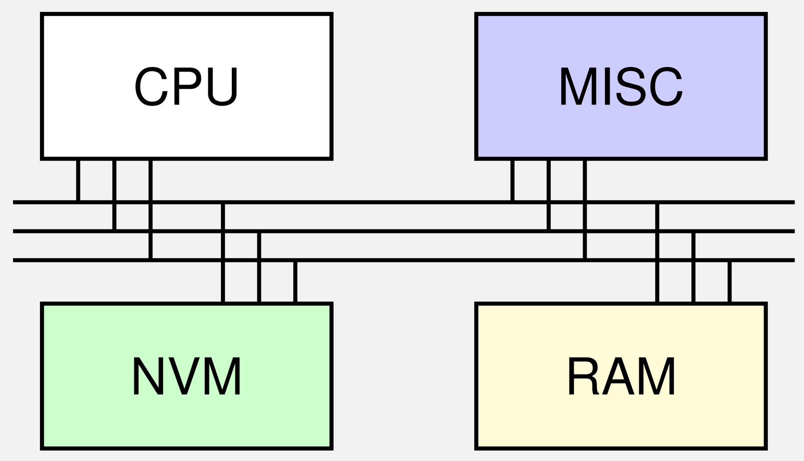
$$g = \arg\min_{f \in \mathcal{F}} L(f, \mathcal{D}) + \lambda R(f)$$

Machine Learning & Emerging Memory

Goal Identify resource saving potentials of non-volatile memories to enable architecture-aware learning algorithms

Non-volatile memories (NVM)

- Slow write, but fast read
- Only infrequent / no refresh required
- Potential drop-in replacement for DDR



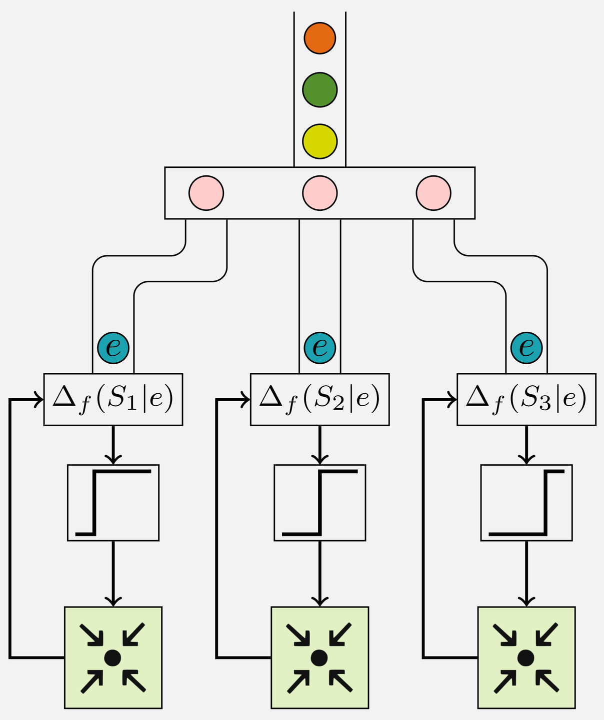
Potential benefits for ML

- Apply ML in heavily resource restricted environments, e.g. smart bins
- Faster and more efficient model learning

Central question How to utilize NVM?
For example, when to use NVM and DDR?

$$\theta^{(t+1)} = \theta^{(t)} - \eta^{(t)} \sum_{i=1}^{N} \nabla \ell(f_{\theta^{(t)}}, x_i, y_i)$$

Data Aggregation and Sampling



Goal Extract representatives from stream

$$S^* = \arg\max_{S \subseteq P(V), |S|=k} f(S)$$

where f is a sub-modular function.

Approach Apply Sieve-Streaming

- Add element if gain exceeds threshold
- Each Sieve has its own threshold
- Guarantees $1/2 - \varepsilon$ approximation by using $\mathcal{O}(\log k/\varepsilon)$ sieves

For example

$$f(S) = \log\det(\Sigma_S)$$

So far Bounding $\log\det(\Sigma_S)$ leads to fewer sieves [Buschjaeger; Morik; Schmidt; IOTStreaming@ECMLPKDD; 2017]

Open questions

- How can we use summaries, e.g. for concept drift detection? B3 C3
- Can we merge/delete elements from a summary?
- What is the relationship with coresets? A2

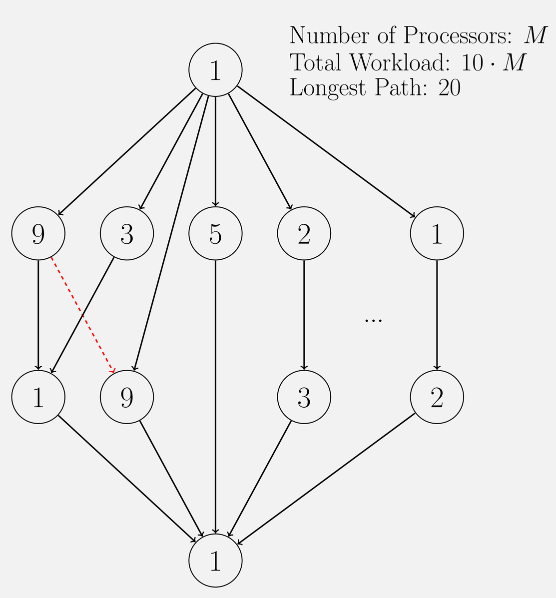
Representation, Execution, and Dependency of Learning

Goal Derive scheduling strategies for classes of ML models

- **Probabilistic guarantees** for both timing behaviour and statistical performance
- Respect precedence constraints using Dependency Graphs (DGs)
- Flexible DG construction and scheduling

Two orthogonal approaches during schedule design

- Start from DG with the best learning output, and remove constraints
- Start from DG with the minimum required learning output, and add constraints



Preliminary results Probability of deadline misses for multi-mode tasks with independent probability [v.d.Brueggen; Piatkowski; Chen; Chen; Morik; ECRTS; 2018]

Open problems

- Probabilistic timing guarantees for dependent random variables
- Dependent execution times in probabilistic graphical models
- Flexible precedence-constraints in scheduling and ML

Artificial Intelligence Group

CS 12 computer science 12