Master thesis

Hardware accelerated embedded on-chip training for reinforcement and machine learning algorithms

The usage of embedded hardware accelerators, such as field programmable gate arrays (FPGAs), has become increasingly popular in the field of machine learning (ML). Traditionally, a ML model is trained offline using dedicated hardware and then deployed on an embedded system for inference [1], [2]. Goals cover energy efficient model execution, low latency, and real-time capabilities. However, the training of ML models on embedded hardware accelerators is still a challenging and largely unexplored task due to complex training algorithms and the lack of established rapid prototyping tools [3], [4]. Such an accelerated training can exhibit compelling advantages for task with low latency training requirements, such as reinforcement learning (RL) control of highly dynamic systems (e.g., electric drives or power electronics). In this domain, edge learning-based RL have been already proposed to utilize the fast inference capabilities of embedded hardware accelerators for real-time control tasks while utilizing dedicated hardware (including standard training routines from PyTorch, JAX,...) for the training procedure [5]. However, this approach comes with the drawback of communication overhead and delays negatively affecting the RL training. Hence, this thesis should explore the potential of training RL policies directly on a hardware accelerated embedded device such that both the training and inference can be executed locally on the same chip.

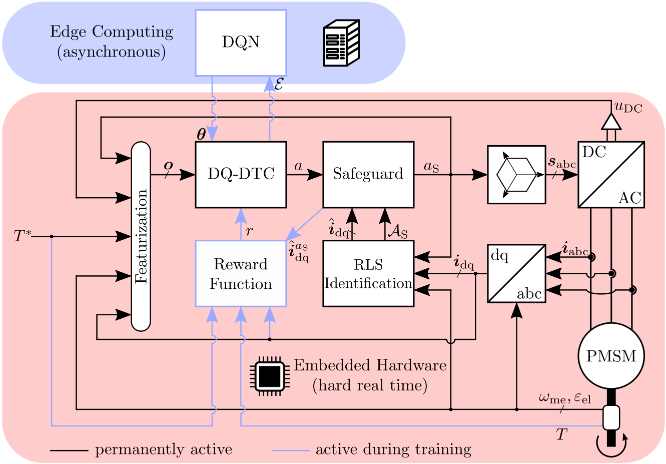

Figure 1: Distributed edge learning for electric drive control [5].

Key research questions:

- Can we utilize embedded hardware accelerators for training ML/RL models on-chip?
- Is the deployment of training algorithms automatable in a rapid prototyping sense?
- What are limitations and challenges of on-chip training on embedded hardware accelerators?

Necessary requirements:

- Finished course work on embedded systems and machine learning
- Solid skills in scientific programming languages (e.g., Julia, JAX, PyTorch)

WP 1: Literature research **[3 weeks]**

Scanning the scientific literature for relevant publications and patents related to training of machine learning / reinforcement learning algorithms and associated IP cores on embedded hardware accelerators is the first step. Moreover, relevant (open-source) software work in the field should be considered. This also includes the identification of relevant keywords as part of the search strategy. Relevant work will be stored in a literature review software (e.g., JabRef) and summarized in the thesis.

WP 2: Baseline implementation **[7 weeks]**

Based on the previous research findings, an IP implementation for training machine learning / reinforcement learning algorithms on an embedded hardware accelerator should be utilized and adapted towards a baseline implemention

WP 3: Critic learning **[7 weeks]**

An important part of many RL algorithms is the training of a critic, that is, an approximate function (typically an artificial neural network) that estimates the value function of the control policy. The value is the discounted long term reward and encodes the usefulness of a certain state-action pair. The critic can be used as a standalone decision-making agent as part of finite control set problems or is utilized as part of an actor-critic architecture to update an explicit control policy (actor). The critic training represents a classical supervised learning problem based on some experience tuples obtained from interacting with the plant environment. Hence, the goal of this WP is to extend the baseline implementation from WP2 to a full-fledged critic training algorithm.

WP 4: Empirical test **[3 weeks]**

Since a special focus of this thesis is on fast online training and inference, the hardware accelerated algorithms should be deployed on an embedded system for software-in-the-loop tests. As a benchmark, dedicated compute hardware (CPU, GPU) should be used in order to evaluate the performance in terms of training time and power consumption.

WP 5: Documentation **[3 weeks]**

All work packages should be reported in a structured way within the thesis. A LaTeX template should be used for this purpose: `https://github.com/IAS-Uni-Siegen/thesis_latex_template`. Writing instructions can be found within the provided template source files. Based on the previous empirical findings, conclusions should be drawn, and future research directions should be outlined.

Gantt chart

The planned timetable is shown in the Gantt diagram below.

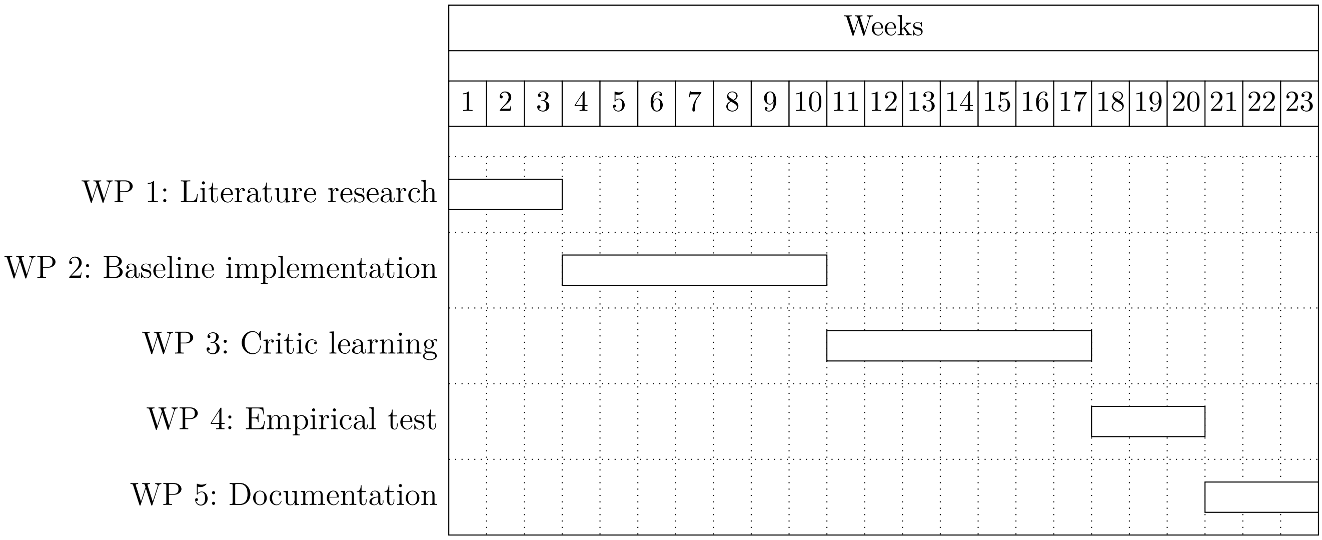

Figure 2: Gantt chart for the thesis.

References

[1] A. Shawahna, S. M. Sait, and A. El-Maleh, "Fpga-based accelerators of deep learning networks for learning and classification: A review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019.

[2] A. G. Blaiech, K. B. Khalifa, C. Valderrama, M. A. Fernandes, and M. H. Bedoui, "A survey and taxonomy of fpga-based deep learning accelerators," *Journal of Systems Architecture*, vol. 98, pp. 331–345, 2019.

[3] N. Sutisna, A. M. R. Ilmy, I. Syafalni, R. Mulyawan, and T. Adiono, "Farane-q: Fast parallel and pipeline q-learning accelerator for configurable reinforcement learning soc," *IEEE Access*, vol. 11, pp. 144–161, 2022.

[4] C.-W. Hu, J. Hu, and S. P. Khatri, "Td3lite: Fpga acceleration of reinforcement learning with structural and representation optimizations," in *International Conference on Field-Programmable Logic and Applications (FPL)*, 2022, pp. 79–85.

[5] M. Schenke, B. Haucke-Korber, and O. Wallscheid, "Finite-set direct torque control via edge-computing-assisted safe reinforcement learning for a permanent-magnet synchronous motor," *IEEE Transactions on Power Electronics*, vol. 38, no. 11, pp. 13 741–13 756, 2023.