Abstract

Hardware-Software Co-Design of Actor-Critic Reinforcement Learning for Efficient and Flexible Edge Platform

Edge devices increasingly require autonomous decision-making capabilities to support emerging applications in robotics, smart sensors, unmanned inspection systems, and embedded cyber-physical systems operating without cloud connectivity. As these systems evolve toward higher autonomy and responsiveness, reinforcement learning (RL) becomes an appealing mechanism for enabling intelligent on-device behaviors. However, the practical deployment of RL—especially Actor–Critic methods—on low-cost embedded hardware is constrained by limited computational throughput, strict power budgets, real-time latency requirements, and restricted memory bandwidth. Actor–Critic algorithms rely on repeated cycles of neural network inference, gradient computation, and frequent weight updates, all of which generate substantial computational load and dense memory traffic. These operations execute inefficiently on embedded CPUs and often exceed the energy envelope of low-power edge GPUs. This motivates the development of tightly optimized hardware–software co-design strategies that distribute computation intelligently between programmable logic and the processing system in heterogeneous FPGA platforms.

To address these challenges, this thesis first develops a comprehensive hardware–software co-design framework for efficient single-agent Advantage Actor–Critic (A2C) training on a resource-constrained FPGA platform. Computationally intensive neural network operations—including matrix multiplications, nonlinear activations, and structured gradient propagation—are offloaded to programmable logic using High-Level Synthesis (HLS) generated feedforward (FF) and gradient computation engines designed with pipeline parallelism and on-chip dataflow optimizations. Meanwhile, the processing system, programmed through the PYNQ framework in Python, manages environment simulation, trajectory collection, action selection, loss computation, and weight updates. This division of labor preserves compatibility with high-level deep learning frameworks such as PyTorch while exploiting FPGA parallelism for compute-dominant operations. The result is a flexible yet energy-efficient training platform that supports multiple experimental configurations without hardware resynthesis.

Beyond feasibility, the single-agent architecture provides detailed insight into how hardware–software partitioning influences performance, scalability, and energy consumption. A shared feed-forward IP core processes both Actor and Critic networks, enabling significant LUT and DSP savings by reusing computations rather than duplicating hardware. Backpropagation is partially accelerated but coordinated through the processing system, reducing hardware complexity at the cost of additional communication overhead. Experimental profiling on CartPole-v1, Acrobot-v1, and LunarLander-v2 demonstrates strong real-time performance and superior energy efficiency compared to CPU and GPU baselines. However,

the study also identifies several architectural bottlenecks that become more pronounced as model complexity increases. Processing system–programmable logic communication, particularly during activation and gradient transfers, emerges as a dominant source of latency. Additionally, sequential weight updates executed on the processing system limit throughput when gradient frequency increases. These findings reveal a fundamental trade-off between hardware simplicity and scalable training performance in embedded reinforcement learning systems.

Motivated by these observations, the second major contribution of this thesis introduces a lightweight, hardware-aware meta-optimizer integrated directly into programmable logic to improve learning stability and reduce total training energy. Conventional optimizers such as Adam and RMSProp rely on static hyperparameters tuned offline and require repeated CPU-side monitoring and adjustment. This creates control overhead and slows responsiveness in resource-constrained environments. To overcome these limitations, the proposed meta-optimizer adopts a hardware-centric design inspired by meta-learning principles but engineered for minimal resource overhead. The optimizer continuously monitors gradient magnitude trends, reward progression, and training stability indicators, using these signals to infer the current learning state. A finite-state machine (FSM), synthesized directly in programmable logic, governs adaptive transitions between learning modes such as aggressive descent, stability correction, and cautious refinement. Each state dynamically adjusts key hyperparameters, including learning rate and update scheduling. These adjustments occur with deterministic, cycle-level latency and do not require processing system intervention.

The hardware-embedded FSM meta-optimizer provides an efficient mechanism for stabilizing on-device A2C training. By relocating adaptive control logic into programmable logic, it reduces processing system overhead and enables immediate responses to instability events. Experimental results demonstrate smoother convergence trajectories, reduced oscillatory updates, and measurable reductions in total training energy. Importantly, these improvements are achieved with minimal additional hardware resource consumption, validating the practicality of embedding optimization intelligence directly into FPGA logic.

Building on insights from the single-agent architecture and meta-optimizer, the final part of this thesis develops a scalable multi-agent reinforcement learning framework tailored for low-cost FPGA platforms. While multi-agent reinforcement learning can significantly accelerate convergence through parallel exploration and variance reduction, naive hardware replication of neural network accelerators per agent is infeasible on resource-limited devices. Instead, this thesis proposes a shared-computation architecture in which all agents utilize a single, deeply pipelined feed-forward engine through time-multiplexed scheduling.

Agent observations stored in DDR memory are streamed into programmable logic using optimized AXI transfers. A scheduling mechanism assigns computation slots to agents in a structured round-robin manner, ensuring fair and deterministic servicing without duplicating hardware resources. Gradients are computed sequentially for each agent using a shared computational core and immediately transferred to the processing system for parameter updates. This strategy maintains controlled hardware growth as the number of agents increases while preserving stable learning dynamics. Experimental evaluation on CartPole and LunarLander confirms that increasing agent count accelerates convergence and improves reward stability

without proportional increases in resource utilization.

Overall, this thesis presents a unified, flexible, and energy-efficient FPGA-based platform for single-agent and multi-agent Actor–Critic reinforcement learning. Through hardware-accelerated neural computation, step-independent memory-efficient backpropagation design, a tightly integrated FSM-based meta-optimizer, and a shared multi-agent scheduling architecture, the proposed framework addresses critical scalability and efficiency challenges in embedded reinforcement learning. The work demonstrates that hardware–software co-design is essential for enabling autonomous, real-time, and energy-aware reinforcement learning directly on low-cost FPGA platforms. Beyond the specific implementations presented, this thesis introduces architectural principles and design methodologies that can guide future development of FPGA-based reinforcement learning accelerators and edge intelligence systems operating under strict resource constraints.

This dissertation consists of six chapters. Chapter 1 introduces the motivation for deploying reinforcement learning on edge devices, reviews existing hardware acceleration limitations, and outlines the research objectives. Chapter 2 presents the theoretical foundations of reinforcement learning and the Actor–Critic framework. Chapter 3 describes the hardware–software co-design for single-agent A2C training and evaluates architectural trade-offs. Chapter 4 introduces the FSM-based hardware meta-optimizer and analyzes its impact on stability and energy efficiency. Chapter 5 extends the system to a scalable multi-agent framework based on shared feed-forward computation and sequential gradient scheduling. Chapter 6 concludes the dissertation and discusses future research directions in FPGA-based reinforcement learning and adaptive optimization.