

| | |
|-------------------------|---|
| 1. Record Nr. | UNINA9910760275803321 |
| Autore | Pasricha Sudeep |
| Titolo | Embedded Machine Learning for Cyber-Physical, IoT, and Edge Computing : Software Optimizations and Hardware/Software Codesign // edited by Sudeep Pasricha, Muhammad Shafique |
| Pubbl/distr/stampa | Cham : , : Springer Nature Switzerland : , : Imprint : Springer, , 2024 |
| ISBN | 9783031399329 3031399323 |
| Edizione | [1st ed. 2024.] |
| Descrizione fisica | 1 online resource (481 pages) |
| Altri autori (Persone) | ShafiqueMuhammad |
| Disciplina | 006.22 |
| Soggetti | Embedded computer systems Electronic circuits Cooperating objects (Computer systems) Embedded Systems Electronic Circuits and Systems Cyber-Physical Systems |
| Lingua di pubblicazione | Inglese |
| Formato | Materiale a stampa |
| Livello bibliografico | Monografia |
| Nota di contenuto | Intro -- Preface -- Acknowledgments -- Contents -- Part I Efficient Software Design for Embedded Machine Learning -- Machine Learning Model Compression for Efficient Indoor Localization on Embedded Platforms -- 1 Introduction -- 2 Background and Related Work -- 3 CHISEL Framework -- 3.1 Data Preprocessing and Augmentation -- 3.2 Network Architecture -- 3.3 Model Compression -- 4 Experiments -- 4.1 Evaluation on UJIIndoorLoc Dataset -- 4.2 Evaluation on Compression-Aware Training -- 5 Conclusion -- References -- A Design Methodology for Energy-Efficient Embedded Spiking Neural Networks -- 1 Introduction -- 1.1 Overview -- 1.2 Design Constraints for Embedded SNNs -- 2 Preliminaries -- 2.1 Spiking Neural Networks (SNNs) -- 2.2 Spike-Timing-Dependent Plasticity (STDP) -- 3 A Design Methodology for Embedded SNNs -- 3.1 Overview -- 3.2 Reduction of SNN Operations -- 3.3 Learning Enhancements -- 3.4 Weight Quantization -- 3.5 Evaluation of Memory and Energy Requirements -- 3.6 Employment of Approximate DRAM -- 4 Experimental Evaluations |

-- 4.1 Classification Accuracy -- 4.2 Reduction of Memory Requirement -- 4.3 Improvement of Energy Efficiency -- 4.4 Impact of Approximate DRAM -- 5 Conclusion -- References -- Compilation and Optimizations for Efficient Machine Learning on Embedded Systems -- 1 Introduction -- 2 Background and Related Works -- 2.1 Efficient DNN Designs -- 2.2 Efficient Accelerator Designs and DNN Mapping Methods -- 2.3 Efficient Co-Design Optimization -- 3 Efficient Machine Learning Model Designs -- 3.1 The ELB-NN -- 3.1.1 Hybrid Quantization Scheme -- 3.1.2 Hardware Accelerator for ELB-NN -- 3.2 The VecQ -- 3.2.1 Quantization with Vector Loss -- 3.2.2 Framework Integration -- 4 Efficient Accelerator Design and Workload Mapping -- 4.1 DNNBuilder -- 4.1.1 An End-to-end Automation Flow -- 4.1.2 Architecture Novelties.

4.1.3 State-of-the-art Performance -- 4.2 PyLog: A Python-Based FPGA Programming Flow -- 4.2.1 PyLog Flow Overview -- 4.2.2 PyLog Features -- 4.2.3 PyLog Evaluation Results -- 5 Efficient Optimizations -- 5.1 Overview of Hardware-aware Neural Architecture Search (NAS) -- 5.2 HW-Aware NAS Formulation -- 5.3 FPGA/DNN Co-Design -- 5.3.1 The Key to Co-Design: Bundle -- 5.3.2 Progressively Reducing Search Space -- 5.3.3 Evaluation Results -- 5.4 EDD: Efficient Differential DNN Architecture Search -- 5.4.1 Fused Co-Design Space -- 5.4.2 Differentiable Performance and Resource Formulation -- 5.4.3 State-of-the-art Results -- 6 Conclusion -- References -- A Pedestrian Detection Case Study for a Traffic Light Controller -- 1 Introduction -- 2 Related Work -- 2.1 Neural Networks for Pedestrian Detection -- 2.2 Pedestrian Detection on Embedded Systems -- 2.3 Quantization -- 3 Pedestrian Detection Use Case -- 4 Results -- 4.1 Experimentation Setup -- 4.2 No Constraints -- 4.3 Cost Constraints -- 4.4 Cost, Latency, and Precision Constraints -- 4.5 Effect of Resolution and Quantization -- 5 Conclusion -- References -- How to Train Accurate BNNs for Embedded Systems? -- 1 Introduction -- 2 Related Work -- 3 Background on BNNs -- 3.1 Inference -- 3.2 Training -- 4 Classification of Accuracy Repair Techniques -- 5 Overview of Accuracy Repair Techniques as Applied in the Literature -- 5.1 Training Techniques -- 5.1.1 Binarizer (STE) -- 5.1.2 Normalization -- 5.1.3 Teacher-Student -- 5.1.4 Regularization -- 5.1.5 Two-Stage Training -- 5.1.6 Optimizer -- 5.2 Network Topology Changing -- 5.2.1 Scaling Factor -- 5.2.2 Ensemble -- 5.2.3 Activation Function -- 5.2.4 Double Residual -- 5.2.5 Squeeze-and-Excitation -- 6 Empirical Review of Accuracy Repair Methods -- 6.1 Establishing the Design Space -- 6.2 Finding a Good Baseline BNN -- 6.3 Design Space Exploration.

6.3.1 Binarizer (STE) -- 6.3.2 Normalization -- 6.3.3 Scaling Factor -- 6.3.4 Two-Stage Training, Activation Function, and Double Residual -- 7 Discussion and Future Research -- 7.1 Accuracy Gap -- 7.2 Benefit and Cost of BNNs -- 8 Conclusion -- References -- Embedded Neuromorphic Using Intel's Loihi Processor -- 1 Introduction -- 2 Brain-Inspired Spiking Neural Networks -- 2.1 Spiking Neuron Models -- 2.2 Spike Coding Methods -- 2.3 SNN Learning Methods -- 3 Conventional Architectures vs. Neuromorphic Architectures -- 4 Event-Based Cameras -- 5 Applications and Datasets for Event-Based SNNs -- 6 The Loihi Architecture -- 6.1 Neuron Model -- 6.2 Chip Architecture -- 6.3 Second Generation: Loihi 2 -- 6.4 Tools to Support Loihi Developers -- 6.5 SOTA Results of Event-Based SNNs on Loihi -- 7 Case Study for Autonomous Vehicles: Car Detection with CarSNN -- 7.1 Problem Analysis and General Design Decisions -- 7.2 CarSNN Methodology -- 7.2.1 CarSNN Model Design -- 7.2.2 Parameters for Training -- 7.2.3 Parameters for Feeding the Input Data -- 7.3 Evaluation of CarSNN Implemented on Loihi -- 7.3.1 Experimental

Setup -- 7.3.2 Accuracy Results for Offline Trained CarSNN -- 7.3.3 CarSNN Implemented on Loihi -- 7.3.4 Comparison with the State of the Art -- 8 Conclusion -- References -- Part II Hardware-Software Co-Design and Co-Optimizations for Embedded Machine Learning -- Machine Learning for Heterogeneous Manycore Design -- 1
Introduction -- 2 ML-Enabled 3D CPU/GPU-Based Heterogeneous Manycore Design -- 2.1 Related Prior Work -- 2.1.1 3D Heterogeneous Manycore Systems -- 2.1.2 Multi-Objective Optimization Algorithms -- 3 3D Heterogeneous Manycore Design Formulation -- 4 MOO-STAGE: ML-Enabled Manycore Design Framework -- 4.1 MOO-STAGE: Local Search -- 4.2 MOO-STAGE: Meta Search -- 5 Experimental Results -- 5.1 Experimental Setup.
5.2 Comparing the Different Algorithms -- 5.3 Comparison with Mesh NoC-Based Heterogeneous Manycore System -- 6 MOO-STAGE FOR M3D-Based Manycore Systems -- 6.1 MOO-STAGE for M3D Design -- 7 Conclusion -- References -- Hardware-Software Co-design for Ultra-Resource-Constrained Embedded Machine Learning Inference: A Printed Electronics Use Case -- 1 Introduction -- 2 Background on Printed Electronics -- 3 Preliminaries -- 4 Bespoke ML Classification Circuits -- 4.1 Resource-Aware ML Algorithm Selection -- 4.2 Bespoke Classifier Implementation -- 5 Co-Design for Approximate ML Classification Circuits -- 5.1 Approximate MLPs and SVMs -- 5.2 Approximate Decision Trees -- 6 Co-design for Stochastic Neural Network Circuits -- 6.1 Mixed-Signal Stochastic Neuron -- 6.2 Analog Stochastic SNG -- 6.3 Analog Stochastic Activation Function -- 6.4 Hardware-Driven Training -- 6.5 Mixed-Signal Stochastic Inference -- 7 Conclusion -- References -- Cross-Layer Optimizations for Efficient Deep Learning Inference at the Edge -- 1 Introduction -- 2 Preliminaries -- 3 DNN Optimization Techniques -- 3.1 Pruning -- 3.1.1 Fine-Grained Pruning -- 3.1.2 Course-Grained Pruning -- 3.2 Quantization -- 3.3 Knowledge Distillation -- 3.4 Neural Architecture Search -- 3.5 Hardware Approximations -- 4 Cross-Layer Optimization -- 4.1 Methodology -- 4.2 Structured Pruning -- 4.3 Quantization -- 4.4 Hardware-Level Approximations: Impact of Self-Healing and Non-Self-Healing Approximate Designs on DNN Accuracy -- 5 End-to-End System-Level Approximations -- 6 Conclusion -- References -- Co-designing Photonic Accelerators for Machine Learning on the Edge -- 1 Introduction -- 2 Background and Related Work -- 3 Noncoherent Photonic Computation Overview -- 4 CrossLight Architecture -- 4.1 MR Device Engineering and Fabrication -- 4.2 Tuning Circuit Design -- 4.3 Architecture Design.
4.3.1 Decomposing Vector Operations in CONV/FC Layers -- 4.3.2 Vector Dot Product (VDP) Unit Design -- 4.3.3 Optical Wavelength Reuse in VDP Units -- 5 Evaluation and Simulation Results -- 5.1 Simulation Setup -- 5.2 Results: CrossLight Resolution Analysis -- 5.3 Results: CrossLight Sensitivity Analysis -- 5.4 Results: Comparison with State-of-the-Art Accelerators -- 6 Conclusion -- References -- Hardware-Software Co-design of Deep Neural Architectures: From FPGAs and ASICs to Computing-in-Memories -- 1 Introduction -- 2 Hardware-Software Co-design with Neural Architecture Search -- 3 Hardware-Aware Neural Architecture Search for FPGA -- 3.1 Implementation of DNNs on FPGAs -- 3.2 Co-design Framework for FPGAs -- 3.2.1 Problem Statement and Solution -- 3.3 Experiments -- 3.3.1 Search Space Setup -- 3.4 Comparison Results with the Existing NAS Frameworks -- 3.5 Comparison Results with the Existing Architectures -- 3.6 Importance of Co-exploration -- 3.7 Concluding Remarks for NAS-F -- 4 Co-design of Neural Networks and ASICs -- 4.1 Problem Analysis for DNN-ASIC Co-design -- 4.1.1 Major

Components -- 4.1.2 Problem Definition -- 4.2 Co-design Framework for ASIC -- 4.3 Experimental Evaluation -- 4.3.1 Evaluation Environment -- 4.4 Design Space Exploration -- 4.4.1 Results on Multiple Tasks for Multiple Datasets -- 4.5 Concluding Remarks for NASAIC -- 5 Co-design of Neural Networks and Computing-in-Memory Accelerators -- 5.1 Compute-in-Memory Neural Accelerators -- 5.1.1 Device and Its Variations -- 5.1.2 Crossbar Architecture -- 5.1.3 NeuroSIM -- 5.2 Problem Definition -- 5.3 Co-design Framework for CiM -- 5.4 Experiments and Results -- 5.4.1 Experiment Setup -- 5.4.2 Comparison Results to State-of-the-Art NAS -- 5.4.3 Results of Multi-Objective Optimization -- 5.5 Concluding Remarks for NACIM -- 6 Conclusions -- References.
Hardware and Software Optimizations for Capsule Networks.

Sommario/riassunto

This book presents recent advances towards the goal of enabling efficient implementation of machine learning models on resource-constrained systems, covering different application domains. The focus is on presenting interesting and new use cases of applying machine learning to innovative application domains, exploring the efficient hardware design of efficient machine learning accelerators, memory optimization techniques, illustrating model compression and neural architecture search techniques for energy-efficient and fast execution on resource-constrained hardware platforms, and understanding hardware-software codesign techniques for achieving even greater energy, reliability, and performance benefits. Discusses efficient implementation of machine learning in embedded, CPS, IoT, and edge computing; Offers comprehensive coverage of hardware design, software design, and hardware/software co-design and co-optimization; Describes real applications to demonstrate how embedded, CPS, IoT, and edge applications benefit from machine learning.
