EMEURO: A Framework for Data-Driven Software Optimization via Deep Learning

Lawrence McAfee and Kunle Olukotun
Stanford University, Pervasive Parallelism Laboratory (PPL)

♦ Introduction

Problem: The benefits of further microarchitecture innovation have marginalized as processors hit a power wall. ASICs are known to have 10^3 better energy efficiency, but unfortunately it is not possible to design new hardware accelerators for each new and often changing software application.

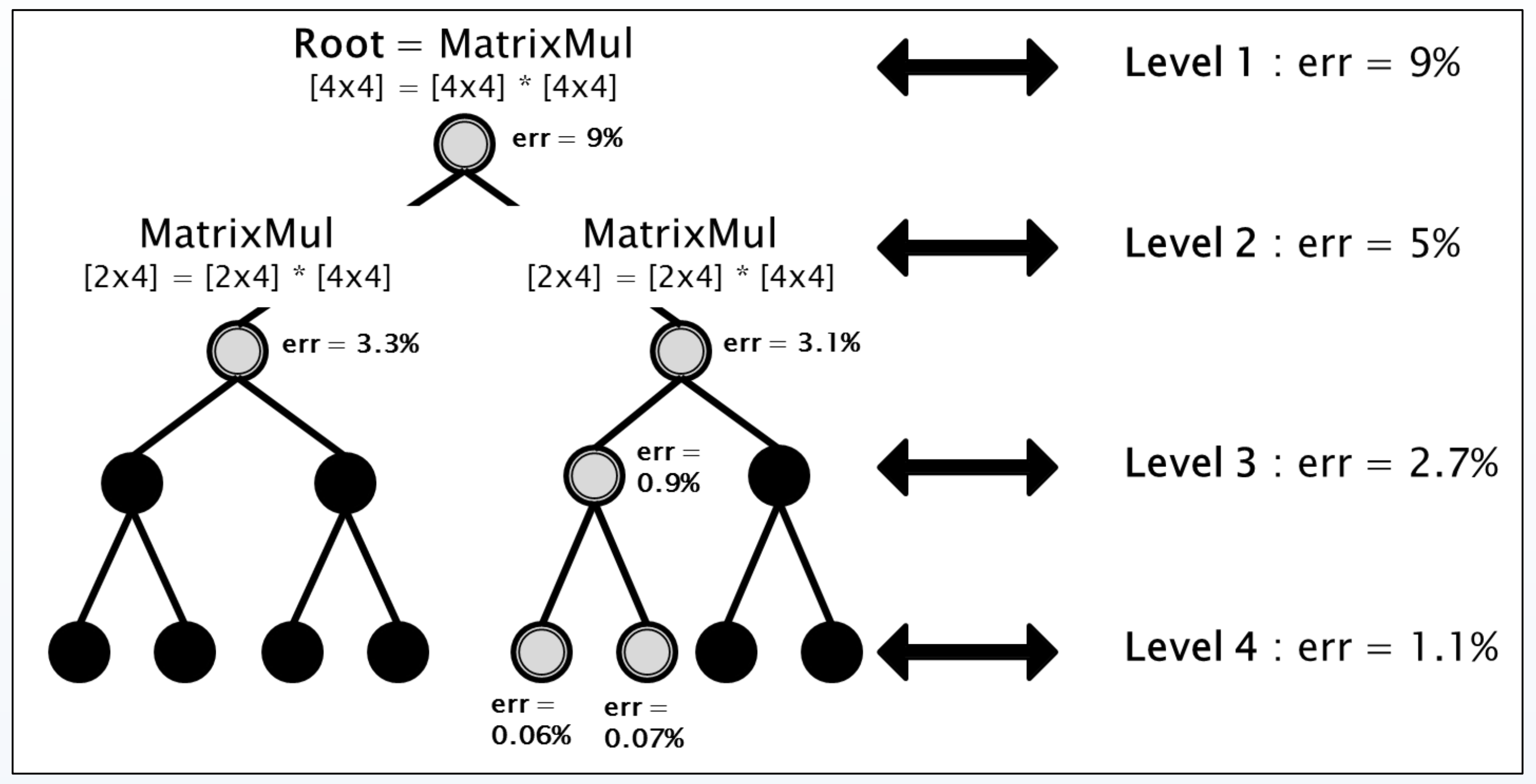
For many applications, while high performance is desired, users do not always require high-precision computations. Applications of this nature can come from many domains, as there is often a speed-versus-accuracy tradeoff a user can choose dependent on specific constraints.

Our work: We introduce EMEURO (pronounced "ee-mure-oh"), an **EM**ulating **NEUR**al network platform for learning models to emulate a wide variety of applications, and then running these models efficiently on a modified GPU architecture. Our main contributions are:

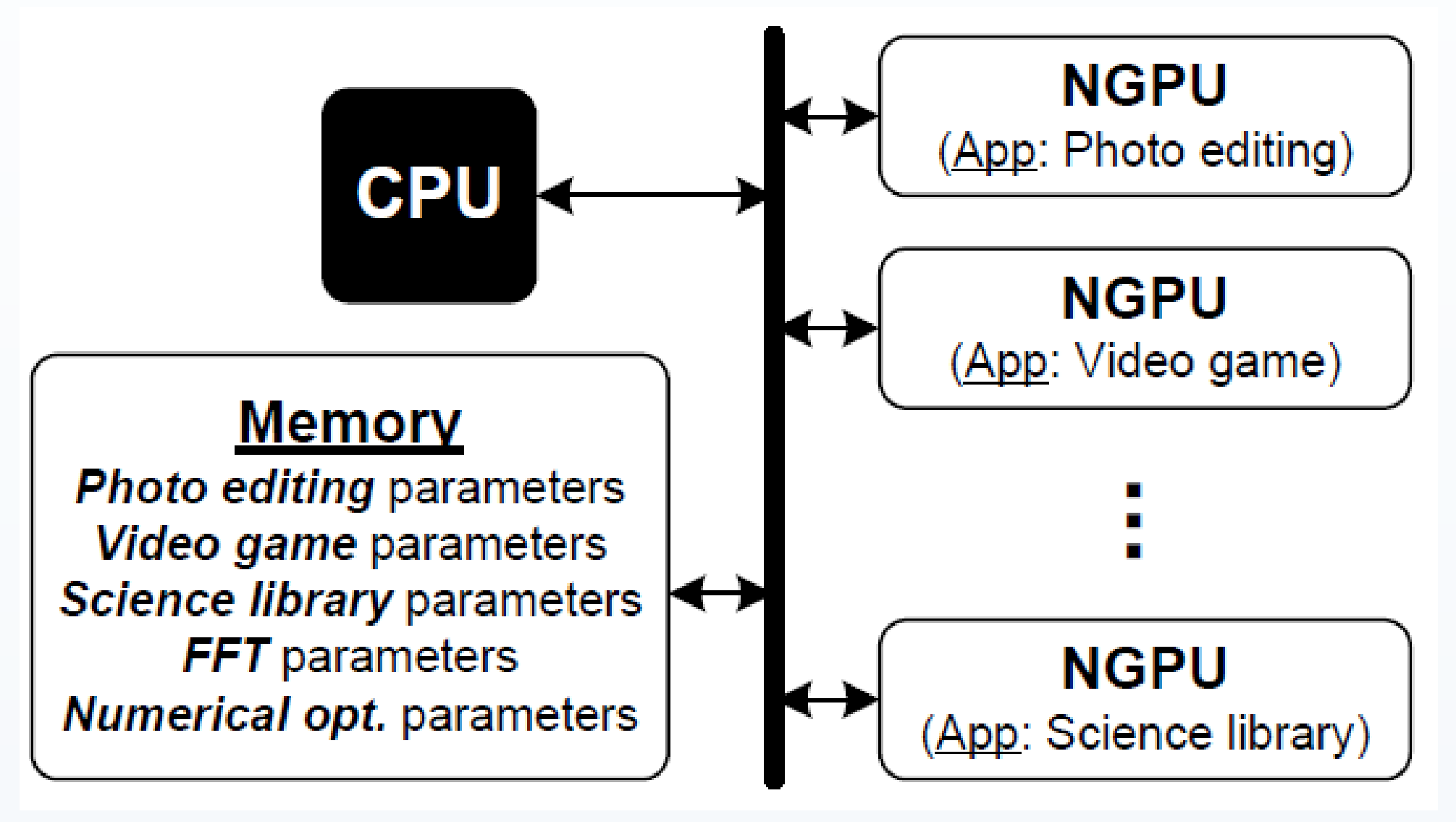
- **EMEURO**, a system for emulating and accelerating applications using neural networks.
- We develop **novel NN compilation techniques**, including: subroutine structure modeling for improved learning; staged training for fine-grain performance tuning at runtime; and clustered emulation for fast online retraining on unfamiliar data.
- We show how to easily **build new applications** in EMEURO, and how to specify exact and approximate regions of code.
- We propose an **augmented GPU architecture** that is optimized to run NNs very efficiently.
- Using real-world data, we show that our system can yield **up to 490X speedup** with less than 2% modeling error.

♦ Compilation/Programming

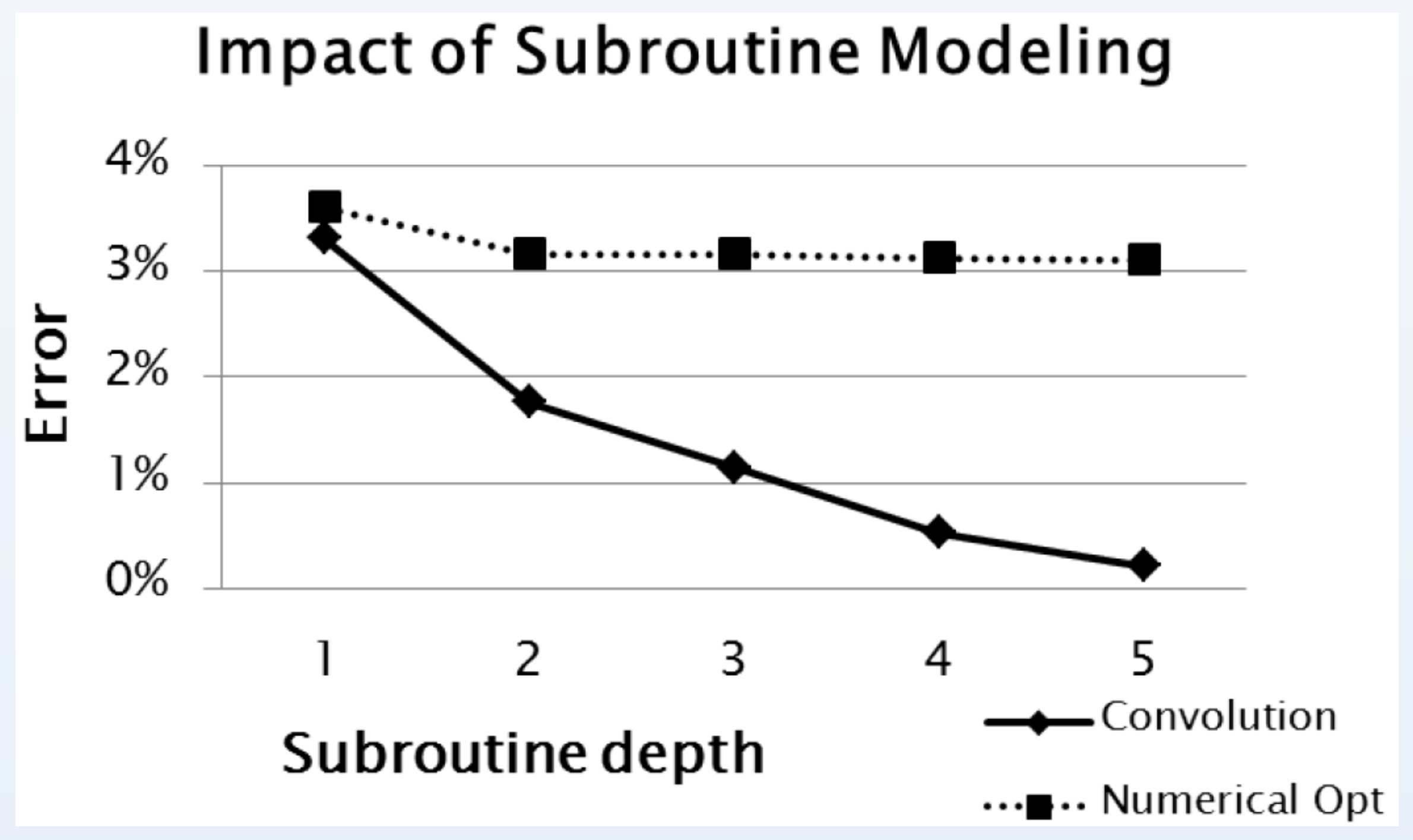
Subroutine utilization

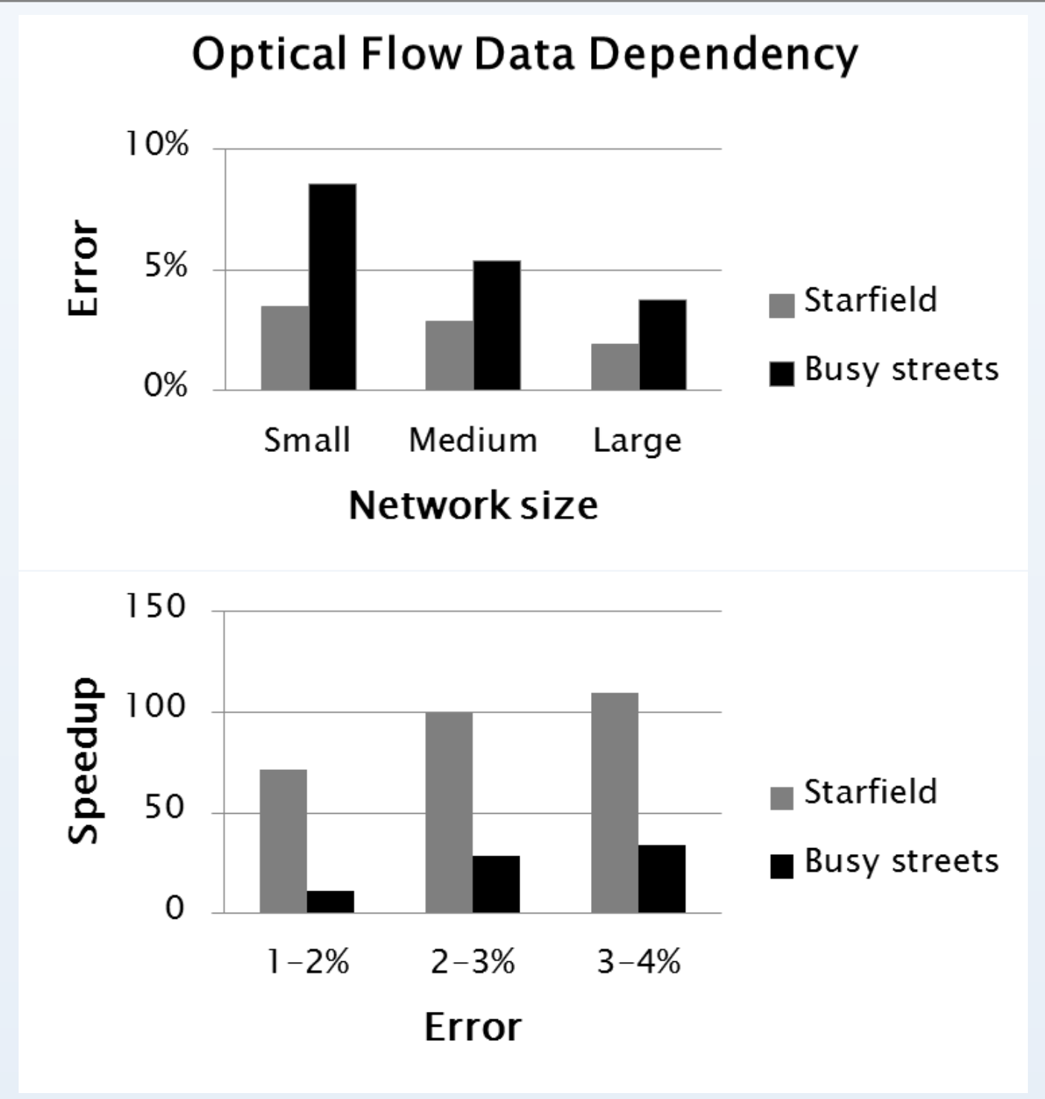


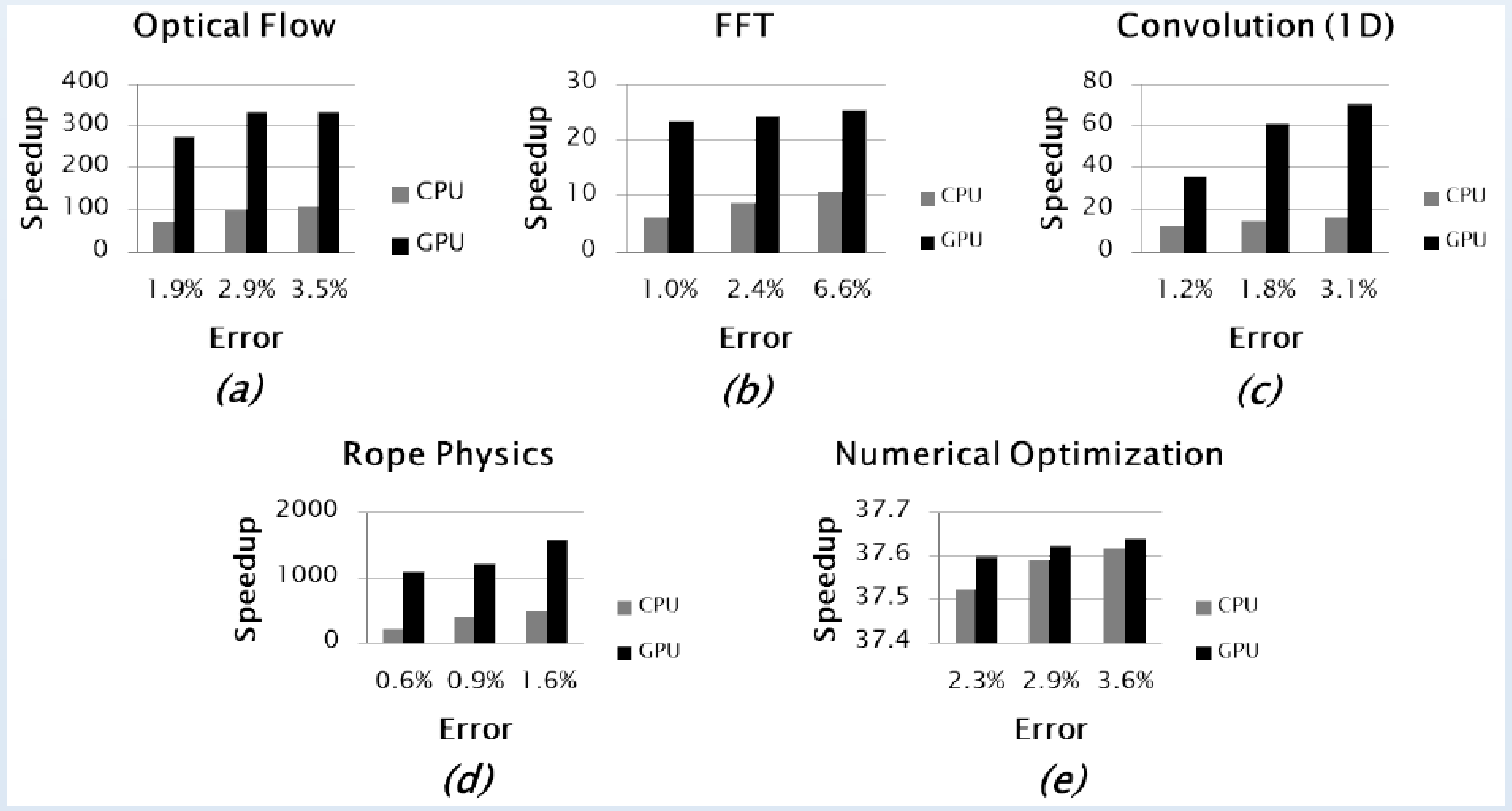
Identical hardware accelerators


CPU
Memory
Photo editing parameters
Video game parameters
Science library parameters
FFT parameters
Numerical opt. parameters
NGPU (App: Photo editing)
NGPU (App: Video game)
⋮
NGPU (App: Science library)

♦ Performance Evaluation







♦ Experimental Methodology

We compare both the approximation error and acceleration of the applications relative to the exact formulation. All timings are performed on the CPU. Tested applications include:

Application	CPU lib	Dimensions
Optical flow (Lucas-Kanade method)	OpenCV [4]	16×16 image patch, 5×5 least squares window
FFT	FFTW [2]	512
Convolution	ALGLIB [1]	512
Rope physics	Nehe Tutorials [3]	100 vertices
Newton's Method	Hand-written	20 vars

The rope physics simulation consists of a 2D mass-spring model swaying side-to-side from a stationary point. This experiments consists of predicting the positions of each vertex in the rope at the next time step using current positions. The Newton's method experiment is completely numerical. Finite differences are used to compute the gradients and the Hessian. The goal of this experiment is to predict the second-order optimization direction using only first-order gradients.

♦ Related Publications

Neural Acceleration for General-Purpose Approximate Programs
Esmaeilzadeh, H., Sampson, A., Ceze, L., Burger, D. *MICRO*, 2010.

A Highly Scalable Restricted Boltzmann Machine FPGA Implementation
Kim, S., McAfee, L., McMahon, P., Olukotun, K. *FPL*, 2009.

♦ Acknowledgements

This research was sponsored by NSF Grant, SHF: Large: Domain Specific Language Infrastructure for Biological Simulation Software, CCF-1111943; the Stanford PPL affiliates, Pervasive Parallelism Lab: Oracle/Sun, NVIDIA, and AMD; and the Stanford Graduate Fellowship.