¡title¿

CONTENTS

I. Introduction

Training a neural network typically requires large-scale data sets, such as ImageNet [?] or MNIST [?], to achieve robust generalization performance. As training involves multiple passes over these data sets and computationally expensive gradient-based optimization, the overall process is highly data-intensive.

To handle data-intensive workloads, data parallelism is required. Data parallelism refers to applying the same arithmetic operations to multiple data elements simultaneously. While modern CPUs support limited forms of data parallelism, their architecture is primarily optimized for low-latency and control-intensive tasks rather than high-throughput parallel computation [?]. GPUs, in contrast, are designed according to a Single-Instruction-Multiple-Data (SIMD) execution model, which enables a much higher degree of data parallelism. Therefore, utilizing GPUs for training neural networks is crucial for the efficient implementation of AI products and solutions. Various hardware vendors provide different software solutions to exploit GPU capabilities.

This work compares the performance of Graphics Processing Unit (GPU) frameworks such as OpenCL [?] and CUDA [?], which enable users to exploit the computational capabilities of GPUs. More precise it compares the solutions based on the benchmark defined in section **??**. To facilitate this comparison, various optimization algorithms are implemented using each framework. This approach is intended to provide insights into the performance characteristics of the different frameworks. In the presented application [?], several optimizers commonly used for training machine learning models are implemented, including Adam [?], AdamW [?], and Stochastic Gradient Descent [?].

CUDA is a software framework for utilizing GPUs, but it is restricted to hardware from the vendor NVIDIA. Furthermore, CUDA is neither standardized nor cross-platform. Nevertheless, it is the most commonly used solution for accessing GPUs in neural network training, as it is implemented in frameworks such as PyTorch, Keras, and TensorFlow [?]. This results in a strong dependence on NVIDIA graphics cards, as other vendors, such as AMD or ARM, are not compatible with the CUDA API [?]. In contrast, OpenCL provides a standardized, cross-platform parallel computing API based on C and C++. OpenCL is open source and maintained by the Khronos Group [?]. Therefore, this report aims to present the performance of non-NVIDIA-dependent solutions in comparison to NVIDIA-specific solutions. However, the application is executed on an NVIDIA GPU.

The different optimizer implementations are integrated into a Deep Convolutional Generative Adversarial Network (DCGAN), whose architecture is based on the PyTorch API Guide [?]. The model architecture, training procedure, and hyperparameters are kept identical across all experiments, with only the optimizer implementation differing between the CUDA- and OpenCL-based approaches.

II. Related Work

In the past other researches already executed the comparision of OpenCL and CUDA as of [**?**], [**?**], [**?**] or [**?**]. Since they are mostly more than 10 years old this another motivation for this report. Nevertheless this section should give some space on their comparision and their results.

III. The Implementation

IV. Performance Benchmarks

V. Conculion

VI. References