University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Tomasz Grześkiewicz
Student no. 394317

Mateusz Kobak
Student no. 385760

Iwona Kotlarska
Student no. 394380

Krzysztof Piesiewicz
Student no. 385996

Dataloading optimization for deep learning on NVIDIA GPUs

Bachelor's thesis
in COMPUTER SCIENCE

Supervisor:
dr Janusz Jabłonowski
University of Warsaw
Faculty of Mathematics, Informatics, and Mechanics
Institute of Informatics

May 2020

Abstract

In this thesis performance of data loading process for deep learning on Nvidia GPUs has been analyzed. The overview of the currently used techniques of optimization are included. Over the course of the work, data loading process has been profiled to identify bottlenecks in the most common use cases in PyTorch and TensorFlow frameworks and chosen ones were mitigated, on either internal level of those frameworks or by creating examples of usage that improve the performance.

Keywords

deep learning, GPU, dataloader

Thesis domain (Socrates-Erasmus subject area codes)

11.3 Informatics, Computer Science

Subject classification

D. Software

Tytuł pracy w języku polskim

Optymalizacja wprowadzania danych na karty graficzne NVIDIA

Contents

Introducion

Topic

For several years the terms *deep learning* and *neural networks* have been getting more and more attention, thanks to the great impact of this technology on a wide variety of applications. *Speech recognition*, *natural language processing*, and *computer vision* are only a few examples of the use cases of the deep learning technology. The increasing demand for deep learning solutions has caused the race for maximizing the performance of computations. From the very beginning of this field, GPUs have been reaching state-of-the-art performance for almost every model and they still remain unbeaten as the best general purpose deep learning computing devices.

The performance of deep learning process depends on many different factors. In this thesis we focus on the data loading process, which is the first step of the model pipeline. By data loading we mean all the operations that are done on the training data before it is processed by the actual neural network. It implies data loading includes not only data preprocessing routines s.a. shuffling, augmenting and different kinds of transformations, but also data transfers (CPU to GPU, GPU to GPU, etc.) and memory alignment. Even though currently in most cases data loading is not a bottleneck of the whole deep learning model, the GPUs are becoming faster so it will likely be the case in the near future.

The process of data loading can be optimized in many ways. Big effort is being put into writing efficient data preprocessing code which runs on GPU - one can mention Nvidia DALI as an example here. Furthermore, deep learning frameworks, s.a. PyTorch and Tensorflow, provide solutions for parallelizing the data loading flow with the training of the neural network. The right use of appropriate features of CUDA libraries is also crucial in terms of the framework's internal code. However, it appears that there are some good practices in using the deep learning framework that might drastically increase the data loading performance.

Over the course of the work we profiled various use cases for PyTorch and TensorFlow. We identified and mitigated some of the most common bottlenecks. We measured achieved performance improvements.

Functionality

Chapter 1

Background overview

Definitions

Definitions used throughout the paper:

- Deep learning - A class of machine learning techniques that exploit many layers of nonlinear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification.
- Host - CPU and memory accessible from there
- Device - single GPU and memory accessible from there
- Dataloading - All the processes that are applied to the data before it's fed to a neural network, including (but not limited to) copying the data from host to device
- Deep learning framework - a set of high level functions that allow users to train neural networks on the device without directly writing device code Iteration - one forward and one backward pass of a single batch of the training examples
- Epoch - one forward and one backward pass of all the training examples
- SIMD - "single instruction multiple data", type of computations in which the same operation needs to be performed on multiple different inputs

Deep Learning

Deep learning is an important technique for feature extraction, pattern analysis and classification. Recent developments in the field allow for solving more and more advanced problems in image classification, object detection, machine translation, language modelling etc.
Computations performed in deep learning are usually SIMD. As a result GPUs, which have multiple simple cores are much better suited for the tasks than CPUs, that have fewer more advanced cores.
There are three main types of deep learning algorithms:

- Supervised learning.
  An algorithm learns from example data and associated target answers. A sample real-world problem is image classification.

- Unsupervised learning.
  An algorithm learns from plain examples without any target answers. It has to find data patterns. For instance, it can be used to determine a class of similar objects. A sample real-world problem is recommendation system.

- Reinforcement learning.
  An algorithm learns from unlabeled data examples. After producing a result it is given feedback with an accuracy of the result. A sample real-world problem is a computer learning to play video games.

Business use-case

Optimized data loading has great commercial value for Nvidia, because it can improve its graphics cards performance without any changes in GPU architecture and give advantage over rival companies' solutions.

Performance of data loading is becoming more and more important since GPUs performance is constantly improving, while single CPU core performance practically stands still. As a result data loading is slowly becoming a bottleneck of machine learning process.

Whole community may benefit from improved data loading, as it will allow more neural network training to be performed in a given period of time and improve throughput of GPUs for models requiring lots of data to be loaded.

Currently used techniques

Over the past few years, the performance of data loading process has been addressed with many optimization techniques. Here we elaborate on the ones that we see as the most important.

- Parallelization – Whilst training runs on device (GPU), the dataloading for the next iteration is performed on host (CPU). Tensorflow guide gives an excellent explanation in []. It is also worth remembering, that operations on CPU can be parallelized across multiple cores.

- Data processing on GPU – Some computations are done much faster on GPU then on CPU. Thus it might be more efficient to execute certain parts of the data processing on the GPU, although we might not fully exploit parallelization this way. We can consider two libraries developed by Nvidia as key examples. Firstly, DALI is a set of image processing tools, executed on the GPU. It is widely used in computer vision systems. Secondly, RAPIDS is data analysis library, which runs on GPU, and is used also in many deep learning ensembles.

- Fast data transfer – There has been a lot effort in both hardware and software to accelerate data transfer from host to device. Currently it is the most efficient to use the CUDA pinned memory technique. It has already been incorporated to both PyTorch and Tensorflow.

- Framework optimizations – Both PyTorch and Tensorflow provide different methods of optimizing the data loading pipeline. These features are often case-specific and one should analyze what works best for the particular model and dataset.

- Storing processed data – It might be efficient to store/cache data. It is not always possible due to lack of memory.