CSCE 614 - Final Report

Aaryan Kothapalli, Michell Brito

12/4/20

1 Abstract

Machine learning has become very popular in recent years which has caused new challenges in computer architecture. Whenever you deal with machine learning, thereâs always going to be a lot of data, and also energy consumption resulting from having to work with so much data. To combat these challenges, there are various ways to minimize the data and energy consumption both at the hardware and software levels. In this paper, we explore the various techniques presented in the paper *hardware for Machine Learning: Challenges and Opportunities*

2 Introduction

Machine learning grew out of Artificial Intelligence as an attempt to have machines solve the large data problem by autonomously learning from them. This gave rise to big data, where ”more data has been created in the past two years than the entire history of the human race”. But to make big data processable, a âsignificant amount of computation is requiredâ. These processing requirements primarily set the limitations to what hardware is viable for Machine Learning. To account for this problem, there are specially designed hardware like TPU, GPUs with large numbers of CUDA cores, and TensorCores to process big data. One example of this is the Nvidia Jetson TX2 which is specifically scaled for Machine Learning applications for developers. There are also cloud applications like Google Collab that provide hardware resources for ML applications. And finally, there are also consumer desktops and laptops with consumer-grade GPUs and CPUs that are also able to run ML applications. The limitations in hardware bandwidth and energy consumption are what primarily differentiate these machines from each other in the context of big data processing.

The Paper Hardware for Machine Learning: Challenges and Opportunities introduce how GPUs and CPUs can perform MAC operations in parallel to reduce the number of multiplications that have to be done in machine learning when generating features. The hardware can do loop unrolling and memory partitioning. Loop unrolling can do multiple calculations per iteration and take larger steps that takes advantage of parallelism. Memory partitioning also takes advantage of parallelism by allowing more instructions to execute concurrently which reduces clock cycles.

Accelerators are introduced as a way to deal with the flow of the dataâs movement which can be optimized by minimizing the access to expensive levels of the memory hierarchy; there are various ones such as output stationary which the outputs are placed in the register file to minimize movement. The paper also suggests that by modifying the machine learning algorithms or hyper-parameters can minimize the computation, data movement and storage requirements such as removing unnecessary weights that are calculated in machine learning algorithms and can help reduce power consumption.

3 Problem

Machine learning is always dealing with large data and performing operations that result in a lot of energy consumption. Whenever a machine learning algorithm is running it has to produce feature extraction, which is used to transform the raw data into meaningful inputs for the given task [1]. The process of feature extraction is very important in regards to machine learning because the quality of the features it extracted can increase the performance of a learning algorithm both in terms of accuracy and computation time. This leads to the problem of having to find a way to minimize the amount of energy that is consumed when doing feature extractions.

Once we have our optimal feature, it is given to us as a vector in which machine learning algorithms use. There are various vectors such as support vector machine (SVM) that can classify the data. To classify the data, SVM and other vectors have to compute the dot product of the features and the weights, this results in a lot of calculations having to be down which increases the energy consumption and the computation time.

Deep Neural Networks (DNN) in machine learning can learn what features are best directly from the data [2]. To learn the features directly from the data, DNN uses multiple layers representing a feature map. These layers are then merged all together to produce the final layer that has the best feature. This method requires a lot of computation due to all the layers.

4 Solution

To solve both the issues regarding energy consumption and computation time in regards to machine learning and computer architecture various things can be done at the hardware and software level such as parallelism, accelerators, testing 3, and testing 4

4.1 Hardware

MAC operations are done during the process of feature extraction and DNN, a way to reduce the energy consumption, and computation time are to do the MAC operations in parallel. The GPU and CPU can take advantage of the temporal locality, which allows the matrix multiplication to be tiled to the storage hierarchy. Loop unrolling helps to optimize the execution time of a program, reducing or eliminating instructions that control the loop. To reduce and eliminate instructions, the compiler modifies the loop so multiple iterations of the loop are executed at once.

Accelerators can also be used to reduce energy consumption and computation time. Accelerators optimize the dataflow of the data to minimize accesses from the memory hierarchy [3]. There are three different dataflows such as Weight stationary (WS), Output stationary (OS). The WS dataflow stores the weights in the register file at the PE, which minimizes the movement cost of the weights, OS stores the output in the register file at the PE, minimizes the cost of thttps://www.overleaf.com/project/5f3eff97c4fcb90001923176he partial sums.

Machine learning models can be trained faster by merely running all operations simultaneously instead of one after the other. Graphics Processing Unit (GPU) can help run these operations parallel because it has more ALUs, control units, and memory cache than CPUs. GPU has a SIMD architecture, which allows it to be able to perform multiple instructions in parallel. Google Colab is a perfect example of using GPU to train machine learning models faster. Google Colab uses a GPU called Nvidia K80 / T4, which contains six Graphics Processing Clusters (GPCs), 36 Texture Processing Clusters (TPCs), and 72 Streaming Multiprocessors (SMs).

4.2 Software

For CNN classification, we know that it needs to produce vectors that require a lot of energy and computational power. A way to reduce the energy and computational power that it requires is to make the weights sparse, this leads to a reduction in the multiplication performed [4]. For feature extraction, the data can be sparse by prepossessing, while for DNNs, the amount of MAC operations and weights can be reduced through pruning. Pruning reduces the size of decision trees by removing parts of the tree that do not provide power to classify instances.

Machine learning algorithms have some parameters that can be adjusted, which are called hyperparameters. Hyperparameters are essential in building accurate models; they're able to find a balance between bias and variance, which prevents the model from overfitting or underfitting. Software such as hyperopt can be used to tune the hyperparameters to optimize the model. The random search parameter tuning algorithm from hyperopt implements a randomized search over parameters. Each setting is sampled from a distribution over possible parameter values; this leads to the best hyperparameters.

4.3 Implementation

- We used two different data sets MNIST (http://yann.lecun.com/exdb/mnist/) and CIFAR10 (https://www.cs.toronto.edu/ kriz/cifar.html). The MNIST dataset contains 60,000 training images and 10,000 testing images of numbers that can be used to train the model to detect a number from an image. The CIFAR10 dataset contains 60,000 32x32 color images of 10 different classes such as airplanes, automobiles, birds, and cats that can train a model to detect what the figures in the picture are.
- We did a convolutional neural network (CNN) to analyze the visual images from the two datasets. To perform CNN, we used a deep learning software called Keras. Using Keras, we normalized the photos, build a linear stack of layers with the sequential model, complied the sequential model, and finally trained the model for ten epochs.
- Now that we had our models, we needed to optimize the hyperparameters by using hyperopt.
- To see the impact that the models have on different machines we tracked

the power consumption, total execution time, CPU usage and GPU usage. We used two tools XRG, and HWinfo64 that helped with tracking these metrics. The metrics allows us compare the hardware performance to efficiently characterize the effects of Machine Learning on these hardware.

- We have four models in total, two of which its hyperparameters are not optimized and two with optimized hyperparameters. The code for these models can be found in our project github: **CSCE_614_Term_Project**

References

[1] Sze, Vivienne, et al. âHardware for Machine Learning: Challenges and Opportunities.â IEEE.org, ieeexplore.ieee.org/document/7993626.

[2] Y. LeCun, Y. Bengio, and G. Hinton, âDeep learning,â Nature, vol. 521, no. 7553, pp. 436â444, May 2015

[3] M. Sankaradas, V. Jakkula, S. Cadambi, S. Chakradhar, I. Durdanovic, E. Cosatto, and H. P. Graf, âA Massively Parallel Coprocessor for Convolutional Neural Networks,â in ASAP, 2009.

[4] A. Suleiman, Z. Zhang, and V. Sze, âA 58.6 mW real-time programmable object detector with multi-scale multi-object support using deformable parts model on 1920Ã 1080 video at 30fps,â in Sym. on VLSI, 2016.