Available online at www.sciencedirect.com

ScienceDirect

ICT Express 4 (2018) 75–80

www.elsevier.com/locate/icte

Slightly-slacked dropout for improving neural network learning on FPGA

Sota Sawaguchi*, Hiroaki Nishi

Department of System Design Engineering, Keio University, 3-14-1 Hiyoshi, Kohoku, Yokohama, Kanagawa 223-8522, Japan

Received 14 February 2018; accepted 10 April 2018
Available online 17 April 2018

Abstract

Neural Network Learning (NNL) is compute-intensive. It often involves a dropout technique which effectively regularizes the network to avoid overfitting. As such, a hardware accelerator for dropout NNL has been proposed; however, the existing method encounters a huge transfer cost between hardware and software. This paper proposes Slightly-Slacked Dropout (SS-Dropout), a novel deterministic dropout technique to address the transfer cost while accelerating the process. Experimental results show that our SS-Dropout technique improves both the usual and dropout NNL accelerator, i.e., 1.55 times speed-up and three order-of-magnitude less transfer cost, respectively.

© 2018 The Korean Institute of Communications and Information Sciences (KICS). Publishing Services by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Keywords: Dropout technique; Mini-batch SGD algorithm; Neural Network; SoC FPGA

1. Introduction

Recent years have seen the rising popularity of neural networks (NNs) for machine learning applications. The assumption of their usage is not limited in Clouds and network edge but also gateways and even mobile devices. For the practical use of NNs, neural network learning (NNL) is indispensable; however, in addition to feed-forward operations, back-propagation is required for NNL, which is why it is compute-intensive. Today, a mini-batch based stochastic gradient descent (SGD) algorithm is typically employed in the back-propagation phase. However, the overfitting problem frequents in NNL. One of the simple and effective regularization methods to prevent overfitting from happening is a dropout technique [1], which stochastically drops out some neurons and trains the sparse network at every weight update (epoch). These two methods are widely used in NNL.

When it comes to accelerating dropout NNL, deterministic dropout techniques have been invented to successfully put aside the dropped out elements and accelerate the process while preserving classification accuracy and overfitting deterrence. Taking power efficiency and design/fabrication cost into account, Field Programmable Gate Arrays (FPGAs) are a good option to further accelerate dropout NNL. Indeed, one existing research by Su et al. [2] deployed the original stochastic dropout on SoC FPGA to achieve the speed-up of dropout NNL, but their strategy has to face a staggering amount of transfer cost between software and hardware, which is attributed to the operation of the dropout algorithm in software. To overcome this issue, we combine the idea of deterministic dropout technique with SoC FPGAs. The major contributions we make in this paper are:

- We propose and implement a novel deterministic dropout technique called Slightly-Slacked Dropout (SS-Dropout) to improve computational delay and data transfer cost of a hardware accelerator for NNL based on mini-batch SGD algorithm on Xilinx Zynq-7020. With the help of interleaved memory system and memory stride access, its hardware design eliminates the problem of the transfer cost expected in [2], along with the full operations of multiply and accumulate (MAC) units to accelerate NNL.
- With 16% of hardware resource overhead, our proposed hardware system yields 1.55 times speed-up compared

* Corresponding author.
E-mail addresses: sawaguchi@west.sd.keio.ac.jp (S. Sawaguchi), west@west.sd.keio.jp (H. Nishi).
Peer review under responsibility of The Korean Institute of Communications and Information Sciences (KICS).

https://doi.org/10.1016/j.icte.2018.04.006
2405-9595/© 2018 The Korean Institute of Communications and Information Sciences (KICS). Publishing Services by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

with the usual NNL accelerator with no SS-Dropout. Also, we confirm three order of magnitude reduction in the transfer cost in comparison with the existing dropout NNL accelerator.

- We verify SS-Dropout's classification accuracy and regularization ability. Experimental results show that classification accuracy might degrade by around 5%, and overfitting could be deterred more effectively as the number of neurons in the hidden layer rises.

2. Related work

As a simple and effective regularization method, the dropout technique has been leveraged in a number of works. Goodfellow et al. [3] use multilayer perceptron for their generative adversarial nets and trains them with the back-propagation and dropout algorithm. [4] applies dropout to the last fully-connected layer of a convolutional neural network (CNN) model for face recognition to achieve comparable verification accuracy while requiring less data compared to the state-of-the-art. Dropout can also be mounted in feed-forward connections in recurrent neural networks (RNNs) to improve the error rates not only in handwriting recognition but also in other applications of RNNs according to Pham et al. [5].

Since stochastic dropout NNL is computationally intensive and time-consuming, some deterministic counterparts are proposed for its acceleration. Wang and Manning [6] proposed fast dropout which samples weighted sums of neural networks from a Gaussian approximation, instead of actually sampling the neurons and training multiple models, which can provide an order of magnitude speed-up at training time. The fast dropout could also yield comparable classification accuracy than the original dropout. The controlled dropout proposed by Ko et al. [7] drops out the units in the matrices in a column-wise or row-wise fashion, so that redundant zero element multiplications in the matrix computation can be eliminated. Their methodology successfully improves training speed and exhibits the comparable regularization ability and classification accuracy, when the number of epochs increases. In this vein, a deterministic dropout could improve performance and sustain the dropout's regularization ability and classification accuracy.

To the best of our knowledge, there is only one research paper by Su et al. [2] that addressed dropout technique on FPGAs. By taking advantage of the reduction in the number of neurons in the dropout network, the NNL process is further accelerated, whereas the transfer cost of weight parameters between hardware and software increases. It is because the dropout algorithm is performed in software at every epoch to select the weight parameters to be transferred to/from hardware in which they are updated. The ideal situation is that one-off mini-batch sized training data are dropped out in software while the corresponding weight data are identified within hardware. Nonetheless, the randomness in the original dropout takes its toll on the "fixed" hardware and deteriorates the transfer cost. As such, considering the comparable performance of deterministic approaches, we devised a novel deterministic dropout technique for FPGAs to diminish the aforementioned transfer cost while maintaining the acceleration.

3. System model and methods

3.1. System overview

Our proposed system is designed on Xilinx Zynq as shown in Fig. 1. Given that the training data are stored in the external DDR memory, mini-batch sized data are randomly selected, SS-Dropped out, and transferred to on-chip BRAMs via AXI CDMA at every epoch. During the process, the SS-Dropout control information that helps stride read/write accessor sift out the interleaved weight parameters within hardware are generated and written into the Neural Central Controller (NCC). This strategy enables reducing the transfer cost. Once all necessary data are prepared and the hardware system is triggered, the Data Switch module separates and switches the data to correct ports of 16 Neural Processing Elements (NPEs) placed in parallel with one another. Note that Fig. 1 illustrates a single NPE which consists of 4 multiply and accumulate (MAC) units, an adder tree, and other arithmetic to support both feed-forward and back-propagation state. Since this paper focuses on classification problem, sigmoid cross entropy in Eq. (1) is used as a loss function with PLAN [8] approximation for the sigmoid activation function. Note that the network output and the corresponding label vector are expressed as $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, and n denotes the batch size. Also, 16-bit fixed data precision is adopted with stochastic rounding mode proposed by Gupta et al. [9].

$$E = -\frac{1}{n}\sum_{n} \boldsymbol{y}^n \log \boldsymbol{x}^n + (1-\boldsymbol{y}^n)\log(1-\boldsymbol{x}^n) \tag{1}$$

3.2. Parallelism exploitation

A mini-batch SGD learning of NNs majorly offers three kinds of parallelism: data-, vector-, and connection-level parallelism. In this section, $f_k^n (n = 1, \ldots, N_b; k = 1, \ldots, N_l)$ denotes the incoming data to a layer, where N_b and N_l stand for the batch size and the number of neurons in the layer, respectively, and w_{ij} a weight coefficient from ith neuron at the layer to jth neuron at the next layer.

Data-level parallelism is defined as how many input neurons can be dealt with at one cycle. Since four MACs reside in our NPE, the four element $f_1^1 \sim f_4^1$ in a single training data vector can be processed in parallel. Secondly, the independence among any training data vector enables vector-level parallelism; in other words, another four elements $f_1^2 \sim f_4^2$ can be tackled by another NPE. Finally, like the input neurons, the counterparts on the output side are independent with one another. Therefore, in addition to $w_{11} \sim w_{41}$, another pair of weight parameters $w_{12} \sim w_{42}$ doubles the connection-level parallelism.

From the aspect of data structure in memory, since four MACs comprise one NPE, 64-bit width memory enables data-level parallelism; another quadruple of the width, i.e., 256-bit width can accommodate four data vectors in a single row to support other two kinds of parallelisms with 16 NPEs, which is implemented in this work.

Download English Version:

https://daneshyari.com/en/article/6925836

Download Persian Version:

https://daneshyari.com/article/6925836

Daneshyari.com