A Green Granular Convolutional Neural Network with Software-FPGA Co-designed Learning

Huaiyuan Chu, Yanqing Zhang *
Department of Computer Science
Georgia State University
Atlanta, GA 30302-5060
chuhy4@gmail.com, yzhang@gsu.edu

Abstract

Different from traditional tedious CPU-GPU-based training algorithms using gradient descent methods, the software-FPGA co-designed learning algorithm is created to quickly solve a system of linear equations to directly calculate optimal values of hyperparameters of the green granular neural network (GGNN). To reduce both CO_2 emissions and energy consumption effectively, a novel green granular convolutional neural network (GGCNN) is developed by using a new classifier that uses GGNNs as building blocks with new fast software-FPGA co-designed learning. Initial simulation results indicate that the FPGA equation solver code runs faster than the Python equation solver code. Therefore, implementing the GGCNN with software-FPGA co-designed learning is feasible. In the future, The GGCNN will be evaluated by comparing with a convolutional neural network with the traditional software-CPU-GPU-based learning in terms of speeds, model sizes, accuracy, CO_2 emissions and energy consumption by using popular datasets. New algorithms will be created to divide the inputs to different input groups for building different GGNNs to solve the curse of dimensionality.

1 Introduction

In recent years, deep neural networks such as a Convolutional Neural Network (CNN) have been effectively used in various applications. However, a major problem is that traditional tedious CPU-GPU-based training algorithms using gradient descent methods take huge amount of training time, generate much CO_2 emissions and waste a lot of energy. For instance, a popular CNN needs a large number of training epochs for very slow hyperparameter optimization. Thus, traditional neural network software systems with very slow hyperparameter optimization algorithms are not suitable for high-speed real-time learning and fast real-time prediction applications. In addition to the long training time problem, the conventional neural networks have the black-box problem (i.e., hyperparameters such as weights are not meaningful). How to build explainable open-box machine learning systems with low CO_2 emissions and low energy consumption is an important long-term problem.

In recent years, new green machine learning (ML) systems have been made to reduce both CO_2 emissions and computational energy consumption. For instance, the AutoML framework for different methods such as neural architecture search (NAS), and automated pruning and quantization is used to build efficient on-device ML systems with low energy consumption and low CO_2 emissions by measuring GPU hours and the estimated CO_2 emission amount CO_2e [1].

*This material is based upon work supported by the National Science Foundation under Grant No. 2234227.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Since CO_2e is proportional to the total computational power p_t: $CO_2e = 0.954p_t$ [2], effectively reducing training times results in greatly reducing both energy consumption and CO_2 emissions.

Currently, popular ML systems running on CPUs and GPUs generate a lot of CO_2 emissions and also waste much energy because (1) tedious traditional training algorithms such as gradient descent algorithms and genetic algorithms take huge amount of time to optimize billions of hyperparameters, and (2) CPUs and GPUs are not green effective and not energy efficient. In summary, an urgent challenge is developing a novel ML system with high-speed non-traditional training algorithms running on the green and energy efficient hardware to significantly reduce both CO_2 emissions and energy consumption.

Based on the successful implementation of the FPGA-based direct linear equation solver [3-5], the high-speed FPGA-based direct linear equation solver can be used to quickly generate optimal hyperparameters in just one epoch for the new green granular neural network (GGNN) in a real-time manner. For example, the Questa*-Intel FPGA Edition Software provides the FPGA design simulation that involves generating simulation files, compiling simulation models, running the simulation, and viewing the results. We use FPGA software simulation systems to implement the high-speed FPGA-based direct linear equation solver. The goal is to develop more effective and faster hardware-based hyperparameter optimization algorithms with a fast direct linear equation solver for training a new GGNN. We develop the novel Green Granular Convolutional Neural Network (GGCNN) with new fast FPGA-based training algorithm to effectively reduce both CO_2 emissions and energy consumption more effectively than the CPU-GPU-based training algorithms.

2 A Building Block: An Efficient FPGA-based GGNN

The GGNN with new fast software-FPGA co-designed learning is designed using granular sets and the software-FPGA co-designed learning algorithm. It uses the software-based learning system to compute the coefficients for a linear system of hyperparameter equations, then uses the fast FPGA-based learning system to optimize the hyperparameters, and finally builds a GGNN model for prediction.

For convenience, an N-record relational database has n numerical input fields x_i for $i = 1, 2, ..., n$, and one numerical output field y. Now the problem is how to build a GGNN using given N records in the relational database. Here, granular sets are used as basic granules in granular partitions of the input variables x_i for $i = 1, 2, ..., n$ and the output variable y. The interval $[a_i, b_i]$ of x_i are partitioned into $m_i - 1$ intervals ($a_{i1} \le x_i \le a_{i2}$, $a_{i2} \le x_i \le a_{i3}$, ..., $a_{i(m_i-1)} \le x_i \le a_{im_i}$). So m_i granules A_{ij} are used to cover the $m_i - 1$ intervals for $i = 1, 2, ..., n$, $j = 1, 2, ..., m_i$. The granules A_{ij} are defined by granular sets such as a fuzzy set [6]. After the above granulation of x_i for $i = 1, 2, ..., n$, there are G data base granules for $G = \prod_{i=1}^{n}(m_i - 1)$. For each data base granule, a GGNN with an output $g(x_1, x_1, ..., x_n)$ is constructed by using two input granular sets covering one interval of x_i and 2^n output granular sets of y. So y has 2^n granular sets B_k for $k = 1, 2, ..., 2^n$.

The granular rule base has 2^n granular IF-THEN rules for one database granule such that $IF\ x_1\ is\ A_{1j_1}\ and\ ...\ x_n\ is\ A_{nj_n}\ THEN\ y\ is\ B_k$ for $j_i \in 1, 2$, $i = 1, 2, ..., n$, and $k = 1, 2, ..., 2^n$.

A database granule has $K = \prod_{i=1}^{n} k_i$ records totally if an input x_i has k_i values for $i = 1, 2, ..., n$ in the database granule, and an output y has K corresponding values y_k for $k = 1, 2, ..., K$. The optimization function for the database granule is to minimize $Q = \frac{1}{2}\sum_{k=1}^{K}[y_k - g(x_{1_k}, x_{2_k}, ..., x_{n_k})]^2$. Based on $\frac{\partial Q}{\partial p_j} = 0$ for the GGNN, we can get a linear system of M-hyperparameter equations for $k = 1, 2, ..., M$ for $M = 2^{n+1}$:

$$T_1^k q_1 + T_1^k q_2 + ... + T_M^k q_M = \psi_k \tag{1}$$

Now we can solve the linear system of M-hyperparameter equations to directly get optimal M hyperparameters q_k of the GGNN for $k = 1, 2, ..., M$.

Based on the successful design of the FPGA-based linear equation solver [3-5], it is feasible to use the same architecture of the FPGA-based linear equation solver to solve equation (1) to get optimized hyperparameters q_k for $k = 1, 2, ..., M$.

The major merits of the granular constructive learning method are (1) quickly optimize parameters using predefined formulas, and (2) discover meaningful granular rules from training data.

We develop the novel GGNN with new fast FPGA-based training algorithm to reduce CO_2 emissions more effectively than the CPU-GPU-based training algorithms. Popular CPUs and GPUs generate much more CO_2 emissions and run less efficiently than the field programmable gate array (FPGA) [7, 8]. For instance, the new FPGA-based massive parallel data processing system can reduce CO_2 emissions by around 50% [8]. FPGA is a light-weight hardware with low CO_2 emissions and low energy consumption [9] for quickly solving a system of linear equations. For example, on a Xilinx Vertex 6 FPGA (200MHz), the minimum latency of the FPGA-based direct linear equation solver was lower than 5 microseconds for a linear system of equations of order 32 [3]. Thus, it is feasible to use FPGA to implement the new FPGA-based training algorithm.

3 Previous Simulations for FPGA-based Learning Methods

Our previous work [10] shows that it is feasible to use FPGA to solve equation (1) quickly by matrix inversion techniques like LUP (LU factorization with partial pivoting). Furthermore, by using HLS (High-Level Synthesis), we can compose codes suitable for FPGA. Experiments also indicate that the performance of FPGA code can outperform MATLAB and Python in the matrix inversion task.

To compare an artificial neural network (ANN) and the GGNN using a fuzzy set (a special granular set), simulations using the 3-input-1-output benchmark function are done [10]. The simulation results show that the $GGNN$ outperforms both the 10-Layer ANN and the 20-layer ANN in terms of the prediction Mean Square Error (MSE), and the prediction Root Mean Square Error (RMSE) for 100, 500, and 1000 training epochs [10].

4 A New GGCNN with Software-FPGA Co-Designed Learning

Since the previous simulations indicate that the new software-FPGA-based learning method is feasible and useful to quickly train the GGNN, the GGNN can be used to build a new machine learning model as a basic building block. A CNN consists of convolutional layers, activation layers, pooling layers, and a classifier such as an MLP. The new GGCNN consists of convolutional layers, activation layers, pooling layers, a new FPGA-based GGNN layer called a FPGA learner, and a hybrid decision model for making a final decision. The new GGCNN with software-FPGA co-designed learning is shown in Fig. 1.

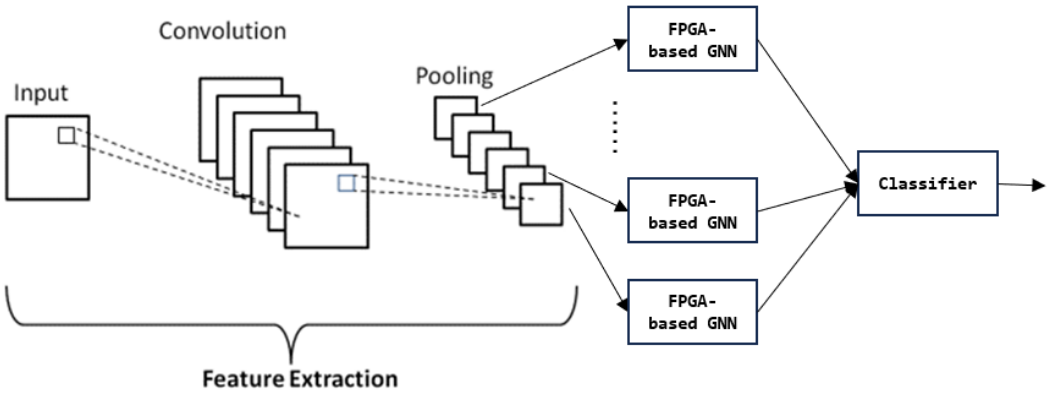

Figure 1: A GGCNN framework with Software-FPGA Co-Designed Learning

Feature maps generated by the last pooling layer are converted to flatten features used by the FPGA-based granular neural network. The outputs generated by the FPGA-based granular neural network are used as inputs by a classifier to make final decisions.

The FPGA-based direct hierarchical hyperparameter optimization algorithm for a GGCNN, a hybrid software-hardware-based algorithm, is given below as Algorithm 1. The linear equation solver can quickly solve a linear system of L hyperparameter equations. A $n \times n$ Feature Map (FM) has $n \times n$ features. N $n \times n$ feature maps FM_p for $p = 1, 2, ..., N$ are generated by the last pooling layer. $K = N \times n \times n$.

for $k = 1$ **to** n **do**

Step 1: Use software to partition $n \times n$ feature map FM_p into M_j $n_j \times n_j$ sub-feature maps for $M_j = l_j \times l_j$, and $n_j < L$).

Step 2: Use software to pre-calculate coefficients for M_j linear systems of hyperparameter equations for $n_j \times n_j$ sub-feature maps): Use software to calculate coefficients such as P_1^{1k}, ..., P_m^{1k}, P_1^{2k}, . . ., P_m^{2k} , P_1^{3k}, . . ., P_m^{3k}, and then calculate $P_1^{1k} = P_1^{1k} + S_1^{1k}$, ..., $P_m^{1k} = P_m^{1k} + S_m^{1k}$, $P_1^{2k} = P_1^{2k} + S_1^{2k}$, ..., $P_m^{2k} = P_m^{2k} + S_m^{2k}$, $P_1^{3k} = T_1^{3k} + S_1^{3k}$, ..., $P_m^{1k} = P_{2^n}^{3k} + S_m^{3k}$ of a linear system of hyperparameter equations.

Step 3: Use the FPGA-based linear equation solver to solve M_j linear systems of hyperparameter equations using M_j $l \times l$ sub-feature maps): Use the FPGA-based linear equation solver to calculate optimal hyperparameters such as (c_i, η_i, and δ_i) for $i = 1, 2, ..., m$ of each linear system of hyperparameter equations. The optimized hyperparameters are used to build new M_j GGNNs with relevant granular knowledge bases with meaningful granular If-Then rules.

Step 4: Use M_j Use the M_j FPGA-based GGNNs to make M_j decisions D_j^p for a new test feature map.

Step 5: Use all individual decisions D_j^p to make a hybrid decision.

end

Algorithm 1: Software-FPGA Co-Designed Learning Algorithm for the GGCNN

5 Conclusions

Initial simulation results indicate that the FPGA equation solver code runs faster than the Python equation solver code. In addition, the GGNN can perform more accurately than a traditional neural network. Therefore, it is feasible to make a novel software-FPGA co-designed GGNN to reduce both CO_2 emissions and energy consumption more effectively than the CPU-GPU-based neural networks. Since FPGA is a high-speed light-weight hardware with low CO_2 emissions and low energy consumption, the FPGA is used to quickly solve a system of linear equations to directly calculate optimal values of hyperparameters of the shallow GGNN. Thus, it is feasible to build the GGCNN using the GGNNs as basic building blocks to solve image recognition problems.

6 Future Works

In the future, the GGCNN with the software-FPGA co-designed learning will be evaluated by comparing with other machine learning models with traditional software-CPU-GPU co-designed learning in terms of speeds, model sizes, accuracy, CO_2 emissions and energy consumption by using popular datasets. New intelligent algorithms will be developed to find out optimal or near optimal sub-spaces in which accurate GGCNN models are built.

A GGCNN with a large number of inputs has the curse of dimensionality. New algorithms will be created to divide the inputs to different input groups that will be used to build different small-size GGCNNs to solve the problem.

We will use different granular sets with different nonlinear membership functions, and then select the best one to improve performance (accuracy, AUC, F1-score, etc.) of the GGCNN.

After the software-FPGA co-designed learning is successful, a special high-speed FPGA hardware based direct linear equation solver as a fast learner will be implemented for building an efficient GGCNN with high classification accuracy to significantly reduce both CO_2 emissions and energy consumption.

References

[1] Y. H. Cai and J. Lin and Y. Lin and Z. Liu and H. Tang and H. Wang and L. Zhu and S. Han, "Enable Deep Learning on Mobile Devices: Methods, Systems, and Applications," ACM Transactions on Design Automation of Electronic Systems (TODAES), Volume 27, Issue 3, Article 20, pp. 1–50, 2021.

[2] E. Strubell and A. Ganesh and A. McCallum, "Energy and Policy Considerations for Deep Learning in NLP," the 57th Annual Meeting of the Association for Computational Linguistics (ACL), 2019.

[3] Z. Jiang and S. A. Raziei, "An efficient FPGA-based direct linear solver," 2017 IEEE National Aerospace and Electronics Conference (NAECON), pp. 159–166, 2017.

[4] L. Miller, "Adaptive Beamforming for Radar: Floating-Point QRD+WBS in an FPGA," 2014.

[5] M. Ruan, "Scalable Floating-Point Matrix Inversion Design Using Vivado High-Level Synthesis," 2017.

[6] L. A. Zadeh, "Fuzzy sets," Information and Control, vol. 8, no. 3, pp. 338–353, 1965.

[7] Using FPGA chip system to reduce the carbon emissions of using web search (https://www.fpgakey.com/technology/details/using-fpga-chip-system-to-reduce-the-carbon-emissions-of-using-web-search).

[8] THALES LEVERAGES NEW TECHNOLOGIES TO BOOST BIOMETRIC MATCHING PERFORMANCE WHILST HAVING ENVIRONMENTAL IMPACT, 20 JAN 2020. (https://www.thalesgroup.com/en/group/journalist/press-release/thales-leverages-new-technologies-boost-biometric-matching).

[9] J. Morss, "FPGAs vs. GPUs: A Tale of Two Accelerators," January 16, 2019. https://www.dell.com/en-us/blog/fpgas-vs-gpus-tale-two-accelerators/

[10] H. Chu and Y.-Q. Zhang, "A Green Granular Neural Network with Efficient Software-FPGA Co-designed Learning," IEEE 22nd International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC'2023), Stanford University, Aug. 19-21, 2023.