Work-in-Progress: Incremental Training of CNNs for User Customization

Mansureh S. Moghaddam[1], Barend Harris[1], Duseok Kang[1], Inpyo Bae[1], Euiseok Kim[1], Hyemi Min[1], Hansu Cho[2], Sukjin Kim[2], Bernhard Egger[1], Soonhoi Ha[1], Kiyoung Choi[1]

[1]Seoul National University
Seoul, Korea

[2]Samsung Electronics, Ltd.
Seoul, Korea

ABSTRACT

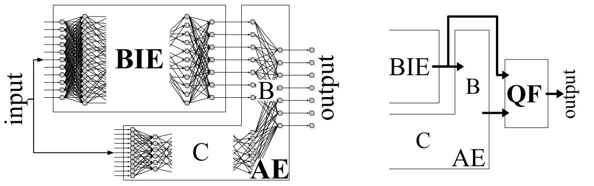
This paper presents a convolutional neural network architecture that supports transfer learning for user customization. The architecture consists of a large basic inference engine and a small augmenting engine. Initially, both engines are trained using a large dataset. Only the augmenting engine is tuned to the user-specific dataset. To preserve the accuracy for the original dataset, the novel concept of quality factor is proposed. The final network is evaluated with the Caffe framework, and our own implementation on a coarse-grained reconfigurable array (CGRA) processor. Experiments with MNIST, NIST'19, and our user-specific datasets show the effectiveness of the proposed approach and the potential of CGRAs as DNN processors.

1 INTRODUCTION

Adoption of DNNs in embedded systems has been limited mainly to classifying a specific input through a trained DNN. Training a DNN requires (1) a well-defined big dataset of training data and (2) a lot of computing power. Limitations of embedded systems, plus the fact that the big dataset is in general not available to the user of the system, render on-device learning a challenging task. Learning without Forgetting (LWF) [3], on-line [5] and incremental [6] learning are a few attempts to deal with the mentioned challenge.

This paper presents a technique that allows on-device personalization of pre-trained large DNNs. We augment an existing *basic inference engine* (BIE) with a small network called *augmenting engine* (AE) that includes a result-aggregation layer as shown in Figure 1a. Inputs are processed in parallel by both networks, and the aggregation layer combines and generates the final result. The BIE and AE are pre-trained by the service provider as usual, but once shipped, only the AE is re-trained on the device. Experiments with handwritten digits and handwritten letters show that the technique is able to improve classification accuracy for individual users from 77% to 91% with a minimal computational overhead.

The potential of CGRAs as DNN accelerators is demonstrated by achieving a 35-fold speedup over a 3-way VLIW processor.



(a) Augmenting the BIE with the AE.

(b) The quality factor.

Figure 1: Specialization of CNNs for user customization.

2 METHODOLOGY

As shown in Figure 1a, the AE consists of two parts: a set of customization layers (`C`) and an aggregation layer (`B`). The basic idea behind this structure is as follows. First, the BIE, trained on a large set of general data, covers the recognition of general data. Block `B` can be used to just pass the recognition results from the BIE to the final output. For user-specific data, block `B` can adjust the results of the BIE to personalize the system by adapting to user-specific data. The control of the behavior of `B` is performed by block `C`. For inputs identified as user data, `C` controls `B` to transform the results of the BIE and enhance the accuracy. Otherwise, `C` lets `B` pass-through the results of the BIE. An incorrect identification in `C` can distort the results of the BIE, resulting in a degradation of the classification accuracy. We include a small decision making logic that determines whether to select the output of the BIE or the AE as the final output based on the quality of the two results. This decision making logic, called quality factor (QF), is shown in Figure 1b.

We assume the following initial training scenario. First, the BIE is trained with a big dataset, then its weights are frozen. We then connect the BIE to the AE, and the entire system is retrained with the same training data. When the system is shipped, the BIE is fixed, but user customization is enabled for the AE (`B` and `C`).

For the implementation of the BIE, we use a dedicated hardware module for energy/performance/cost efficiency. For the AE (both retraining and inference), however, we use a hybrid VLIW/CGRA processor similar to SRP [7], because it provides flexibility, and more importantly, is already present in our embedded platform to perform various other functions. The base configuration is a 32-bit floating-point CGRA with 4x4 heterogeneous processing elements (PE) and a total of 320KB of on-chip data SRAM. Four PEs are connected to the data memory, half of the 16 PEs support floating point operations. To accommodate for the higher bandwidth of DNN processing, four more PEs support memory operations, yielding a maximal theoretical throughput of 8 load/store operations per clock cycle.

CASES '17 Companion, Seoul, Republic of Korea

978-1-4503-5184-3/17/10…$15.00
DOI: 10.1145/3125501.3125519

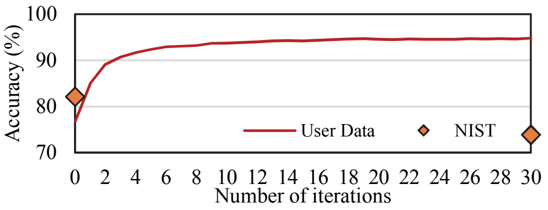


Figure 2: Accuracy of BIE + AE on user data vs. original data after retraining.

Accordingly, the number of banks in the data memory has been increased from four to eight. We have chosen to implement a simple deep neural network framework written in C that does not have any external dependencies in order to support embedded CGRAs. The implementation of certain training functions, especially the gradient computations, have been taken from Darknet [4].

3 EVALUATION

We evaluate the proposed design for recognition of (1) handwritten digits and (2) handwritten letters and digits. For the BIE, we use LeNet-5 [2] and the best performing CNN Committee classifier [1] for (1) and (2), respectively. The AE consists of two pooling and one convolutional layer for block C and one fully connected layer for block B. The QF logic selects the final output based on a simple comparison of the outputs generated by the BIE and AE. Training is performed using the MNIST and NIST training databases, respectively. For the user-specific training, we use the USPS test set and handwritten datasets gathered from 10 users. Each dataset contains 40 subsets (30 for training and 10 for test); one subset contains 62 images, one for each of the 26 lower- and uppercase characters and the 10 digits. For the training, each subset is used as a mini-batch.

Figure 2 shows how the average accuracy for the user dataset increases as the number of iterations (the mini-batch used in each iteration consists of 62 images) increases. Also note how the accuracy for the original dataset decreases. Table 1 lists the results before (pre) and after (post) training with the 10 user datasets for each of the four categories (`all`, `lower`, `upper`, and `digit`). We observe that the proposed system successfully specializes to the handwriting of the individual end users. Untrained, the system achieves an accuracy of only 76.3% for the most general (`all`) alphanumeric character set of user data, but after only five iterations, the average accuracy reaches 92%. The system saturates after around 15 iterations, indicating that an end user will not have to train the system for a long time to achieve good results. More importantly, the accuracy of the proposed system surpasses the BIE by almost 10% after specialization. We also note that specialization taints the output of the entire system for general data (`NIST` column) and all alphanumeric characters by 10%; below we show that the QF can eliminate this effect.

With the QF logic, the system loses a few percent of accuracy on the user datasets, but is able to almost preserve the accuracy on the original data. Results for test on original data for recognizing handwritten digits (MNIST) with and without the QF are shown in Table 2. Results for the NIST dataset are similar.

Table 1: Classification accuracy before and after retraining the system using user-specific datasets.

	all		lower		upper		digit	
Dataset	pre	post	pre	post	pre	post	pre	post
NIST	82.2	73.0	96.2	96.0	88.8	86.9	98.2	96.4
User-01	68.6	87.3	86.5	96.2	95.8	98.5	97.0	99.0
User-02	78.4	97.1	94.2	100	100	100	91.0	100
User-03	78.1	93.6	97.3	98.9	99.2	99.6	98.0	100
...	...	...	...	...	...	...	...	...
User-10	82.1	97.4	95.0	99.2	99.7	100	100	100
Avg	76.3	93.2	92.5	98.9	97.4	99.4	97.5	99.8

Table 2: Quality Factor efficiency on preserving the accuracy on the original dataset after specialization.

	test on user data		test on original data	
Dataset	no-QF	QF	no-QF	QF
USPS	93.78%	92.08%	75.93%	98.28%

The mapping of the proposed AE to the DNN-optimized CGRA using our custom compiler achieves a 35-fold speedup compared to a relatively powerful 3-issue VLIW. At a clock frequency of 500MHz, retraining the system with one user image takes only 2.3 ms on the CGRA compared to 81 ms on the VLIW architecture.

4 CONCLUSIONS

We have introduced a novel network structure comprising a large basic inference engine and a small augmenting engine for user-specialization. The results on handwritten digits and alphanumeric characters show the potential of the proposed system. While state-of-the-art classifiers trained with general data show an accuracy of less than 77% for alphanumeric characters, the proposed structure achieves an average accuracy of 93% with a moderate memory space overhead of around 4% for the weights of the AE. A 35-fold speedup when mapping the augmenting engine to a DNN-optimized CGRA shows the potential of the architecture for DNN processing.

ACKNOWLEDGMENTS

This work was supported in part by grants 21A20151113068 and NRF-2015K1A3A1A14021288 of the National Research Foundation of Korea and by the Promising-Pioneering Researcher Program of Seoul National University in 2015. ICT at Seoul National University provided research facilities for this study.

REFERENCES

[1] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber. 2011. Convolutional Neural Network Committees for Handwritten Character Classification. In *2011 International Conference on Document Analysis and Recognition*. 1135–1139.

[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[3] Zhizhong Li and Derek Hoiem. 2016. *Learning Without Forgetting*. Springer International Publishing, Cham, 614–629.

[4] Joseph Redmon. 2013–2016. Darknet: Open Source Neural Networks in C. http://pjreddie.com/darknet/. (2013–2016).

[5] David Saad (Ed.). 1998. *On-line Learning in Neural Networks*. Cambridge University Press, New York, NY, USA.

[6] Yuan-Yuan Shen and Cheng-Lin Liu. 2016. *Incremental Learning Vector Quantization for Character Recognition with Local Style Consistency*. Springer International Publishing, Cham, 228–239.

[7] Dongkwan Suh, Kiseok Kwon, Sukjin Kim, Soojung Ryu, and Jeongwook Kim. 2012. Design space exploration and implementation of a high performance and low area Coarse Grained Reconfigurable Processor. In *International Conference on Field-Programmable Technology (FPT)*. 67–70.