POSTER: Bridging the Gap between Deep Learning and Sparse Matrix Format Selection

Yue Zhao*, Jiajia Li†, Chunhua Liao‡ and Xipeng Shen*
* North Carolina State University, Raleigh, NC
{yzhao30, xshen5}@ncsu.edu
† Georgia Institute of Technology, Atlanta, GA
jiajiali@gatech.edu
‡ Lawrence Livermore National Laboratory, Livermore, CA
liao6@llnl.gov

***Abstract*—In this work, we conduct a systematic exploration on the promise and challenges of deep learning for the sparse matrix format selection. We propose a set of novel techniques to solve special challenges to deep learning, including input matrix representations, a late-merging deep neural network structure design, and the use of *transfer learning* to alleviate cross-architecture portability issues.**

***Index Terms*—SpMV, sparse matrix storage format, deep learning**

I. Introduction

Sparse matrix vector multiplication (SpMV) is one of the most important kernels in many scientific applications and also often the performance bottlenecks.

One of the most important optimization for SpMV performance is the selection of the proper format to represent sparse matrices in memory. This is a challenging task for programmer since the proper format of a sparse matrix depends on its matrix size, nonzero distribution, architecture characteristics, and so on. It is also challenging to use traditional machine learning due to the difficulties in coming up with the right features of matrices for learning and the complex relations between SpMV performance and the proper format of a sparse matrix,

By treating a matrix as an image, the problem could map to an image classification problem. The success of Deep Neural Networks (DNN) in image recognition suggests the promise of DNN for sparse matrix format selection. However, there are some special challenges on input matrix representation, DNN structure design, and the needs for cross-architecture migrations of the learned models. This paper presents our research results for addressing these challenges. Our work is based on four basic formats CSR, COO, DIA, ELL, which are extensively used in numerous applications. The solution can be extended to more formats.

II. Overview

The overall process consists of four steps: 1) collecting labels by running SpMVs on combinations of the training matrices and the four formats. For each matrix, it labels it with the format with which SpMV runs the fastest, 2) normalizing each of the matrices to one size as required by CNN, 3) designing the structure of CNN, and 4) running the standard CNN training algorithm.

For prediction, a given matrix is first normalized to the fixed size and then fed into the trained CNN, the output nodes give the probabilities for each of the formats to be the best choice.

III. Challenges and solutions

A. Input representation

For various sized matrices to work with CNN, they have to be normalized to a single size. This is called matrix normalization. It is important that the normalization keeps features of the original matrix that are critical.

We start with applying the image scaling method to matrices. It maps non-zero elements in the original matrix to elements in the normalized matrix based on the scale ratio. A region containing one or more non-zero elements becomes 1 in the normalized matrix, and 0 otherwise. This results in a binary matrix.

Scaling keeps the coarse-grained patterns but may lose some subtle but useful info. Thus we introduce a new representation, *density representation*, to complement the binary image representation. Instead of producing zero or one for each region of the original image, it records the number of nonzero entries in a region divided by the region size. We will use both representations as the input of the CNN model.

B. DNN structure design

In our design, we come up with a *late-merging structure*. As Figure 1 illustrates, the structure consists of two separate convolutional networks with each processing the info from one source, and only at the very last stage, the outputs of the two networks are merged as joint features, fed to the fully connected layer for the final output. The two convolution networks can be regarded as processes to extract the critical features from each of the two sources of input information. The final layer combines these features together to make the final prediction.

C. Cross-architecture adaptations

Except for the matrix pattern, the performance of SpMV is also dependent on the machine (e.g., memory bandwidth,

978-1-5090-6764-0/17 $31.00 © 2017 IEEE
DOI 10.1109/PACT.2017.33

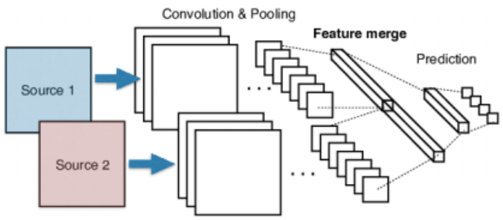

Fig. 1. Proposed late-merging CNN structure.

TABLE I
THE PREDICTION ACCURACY, RECALL, AND PRECISION (B/D: BINARY/DENSITY, DT: DECISION TREE).

Format	Ground Truth	CNN+B/D		DT	
		Recall	Precise	Recall	Precise
COO	667	0.71	0.74	0.53	0.61
CSR	6947	0.94	0.94	0.90	0.88
DIA	894	0.82	0.85	0.83	0.75
ELL	692	0.82	0.80	0.71	0.85
Total / Accuracy	9200	0.90		0.85	

cache size, number of cores). Thus, a prediction model built for one machine will not work well for another.

On the other hand, training a new CNN on a platform includes the collection of labels by rerunning SpMV on each matrix on the new machine and rerunning the CNN training algorithm to determine the appropriate parameters. This process is time consuming.

To efficiently migrate a model across systems, we explore the use of *transfer learning*. We use two methods. The first is called *continuous evolvement* where we initialize a new model with parameters trained on previous machine and continue to train it with new data collected on this machine. The second method is called *top evolvement*, where we fixed all early layers and only retrain the top fully connected layer.

IV. Evaluation

To evaluate the efficacy of the technique, we compare with the state-of-the-art method [1], which involves manually designed set of features of matrices and a decision tree model. In addition, we report the impact of the two methods of *transfer learning*.

Our experiments use a set of 9200 matrices including the 2757 real-world matrices from the SuiteSparse matrix collection [2] and others derived from them.

Table I shows the precision and recall values of four formats of our CNN-based models and the previous Decision Tree-based model (DT) [1], where the CNN model uses 128 × 128 as the size of the representations.

Figure 2 shows the speedup distribution over testing matrices on which the two models give different predictions. The DNN model helps improve the SpMV performance on 86% matrices over the DT model. The SpMVs using the DNN model predicted formats achieve an average of 1.73× and the maximum of 5.2× speedups over those of the DT model.

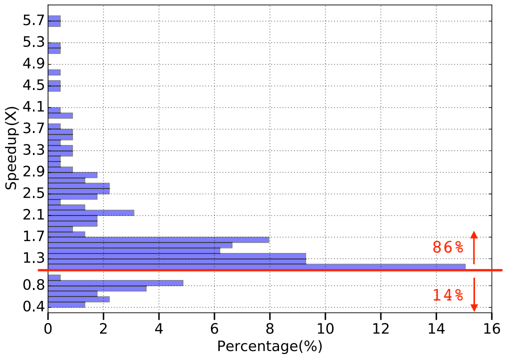

Fig. 2. The speedup with respect to the DT-based prediction.

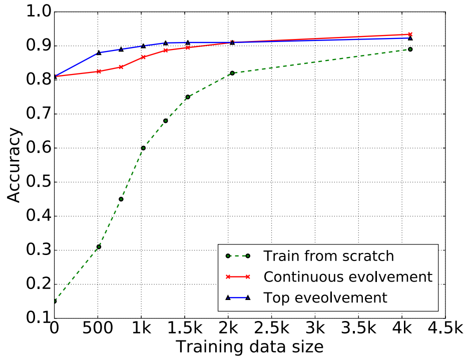

Fig. 3. Prediction accuracies of different retraining methods on a new platform (Intel Xeon E5-4603 to AMD A8-7600 Radeon R7).

Figure 3 illustrates the effect of the two transfer learning methods compared to the training from scratch in terms of prediction accuracy. With "top evolvement", it takes only about a quarter of the time the "from scratch" method takes.

V. Conclusion

We present a systematic exploration on closing the gap between DNN and sparse matrix format selection. The resulting predictive model significantly reduces the prediction errors and brings substantial speedups for SpMV compared to the state of the art techniques. As one of the pioneering studies on bridging the gap between DNN and HPC, this work provides a set of insights that can potentially help the adoption of DNN in solving many other HPC problems.

References

[1] J. Li, G. Tan, M. Chen, and N. Sun, "SMAT: An input adaptive auto-tuner for sparse matrix-vector multiplication," in *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '13. New York, NY, USA: ACM, 2013, pp. 117–126. [Online]. Available: http://doi.acm.org/10.1145/2491956.2462181

[2] T. A. Davis and Y. Hu, "The university of florida sparse matrix collection," *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 1:1–1:25, Dec. 2011. [Online]. Available: http://doi.acm.org/10.1145/2049662.2049663