EPiC Series in Computing

Volume 69, 2020, Pages 355–361

Proceedings of 35th International Conference on Computers and Their Applications

Zero-skipping in CapsNet. Is it worth it?

Ramin Sharifi, Pouya Shiri and Amirali Baniasadi
ECE Department, University of Victoria
raminsharifi@uvic.ca, pouyashiri@uvic.ca, amiralib@uvic.ca

Abstract

Capsule networks (CapsNet) are the next generation of neural networks. CapsNet can be used for classification of data of different types. Today's General Purpose Graphical Processing Units (GPGPUs) are more capable than before and let us train these complex networks. However, time and energy consumption remains a challenge. In this work, we investigate if skipping trivial operations i.e. multiplication by zero in CapsNet, can possibly save energy. We base our analysis on the number of multiplications by zero detected while training CapsNet on MNIST and Fashion-MNIST datasets.

1 Introduction

Machine learning methods have a wide range of applications such as Natural Language Processing (NLP), pattern recognition and computational learning. Deep learning structures were introduced in the late 20th century and originated from studying Artificial Neural Networks (ANNs) [1] which has attracted a lot of focus in recent years [2]–[6]. A Neural Network is constructed using a network of neurons that produce activations and weights needed to be adjusted for the network to learn a specific pattern in data [1]. Deep Neural Networks (DNNs) use a high number of layers to learn features of different depths from the input data.

Convolutional Neural Networks (CNNs) are a type of DNNs that performs effectively on cases in which the input data is two-dimensional, e.g., images [1]. These networks include several types of layers. One of the layers used in CNNs is the Pooling layer which is used to create different receptive fields to obtain richer semantic information from the feature maps. This layer reduces the size of the previous layer and results in losing some information. Pooling is a convolutional layer and is responsible for keeping some values per region of the feature map depending on its type.

Capsule Network (CapsNet) is the next-generation neural network and has drawn the attention of researchers in the past few years [7]–[10]. The concept "Capsule" was first introduced in [11] but there was no practical implementation until Sabour et al. introduced "Dynamic routing between Capsules" [12] and presented a practical new network based on capsules called Capsule Network. A capsule is a set of neurons that provide vectors as output (instead of scalars used in CNN) which represent different properties called the instantiation parameters for a type of entity. In contrast to

CNNs, this network considers the relationship between low-level and high-level features while performing the classification. This is done by using vectors to represent features and applying an iterative algorithm referred to as “Routing by Agreement” or “Dynamic Routing”, to find the relationship between vectors of one layer and the output vectors of the next one.

Trivial operations are those operations where the output can be calculated without performing the operation. Examples include multiplication or addition to zero. Skipping trivial operations results in networks requiring less memory footprint and performing faster training and testing. This consequently results in saving time and energy. There are several works on trivial operations in neural networks [13]–[15]. Here we focus on zero-skipping and identifying and skipping redundant multiplications by zero. CapsNet architecture includes two convolutional layers, a weight matrix multiplication and the dynamic routing algorithm to generate the output vectors whose length determines the type of an entity. These vectors are also fed to 3 Fully-Connected (FC) layers to reconstruct the input image. This reconstruction is used as a regularizer for the training process. Figure 1 shows the architecture of CapsNet.

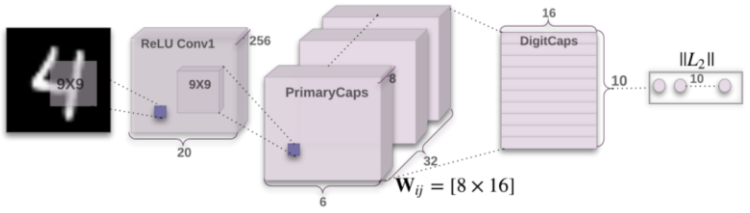


Figure 1. Architecture of Capsule Network [1]

In this work, we investigate the maximum savings achievable by removing trivial operations in CapsNet by analyzing the fraction of zeros present in the CapsNet predictions. These are the zeros generated by the primary capsules on the validation set for different epochs. This investigation serves as a first step in estimating the potential benefits of applying zero-skipping techniques on CapsNet. All experiments are done on MNIST handwritten digits [16] and Fashion-MNIST [17] datasets.

The organization of this paper is as follows. In section 2, we review recent works on skipping trivial operations in literature. In section 3, we explain CapsNet architecture and our methodology. Section 4 includes our experiments and their corresponding results. Section 5 offers conclusions and presents future work.

2 Related Works

Zero-skipping and power aware neural networks have attracted the attention of research community in recent years. In [15], Albercio et al show that on average 44% of Convolutional Neural Network multiplications produce zero. Note that their provided architecture is a modified version of DaDianNao architecture [18]. Albercio et al [15] introduced CNVLUTIN as a modified architecture to skip multiplications by zero in Convolutional Neural Networks. CNVLUTIN improves performance on average by 34%. CNVLUTIN’s best and worst results were on cnnS with 55% and GoogleNet with 24% performance improvement, respectievly. They introduced a zero-free neuron array format which only contains non-zero value neurons to avoid zero multiplication. In order to keep all lanes busy, they use a dispatcher to store neural values to fill empty lanes. CNVLUTIN decouples the original architecture lanes so each lane can perform zero skipping.

In ZeNA [14], Kim et. al were the first to skip trivial computations (also referred to as ineffectual operations) caused by weights and activations. They proposed zero-aware kernel allocation and dynamic work group. They report 4.4 times speed up for AlexNet over fixed-point eyeriss which they use for baseline. Kim et al [14] present ZeNA as a highly scalable solution. In their work, zero-aware kernel allocation sorts the kernel values first and then allocates them to processing elements. By this method, the load is distributed more uniformly. Dynamic work group deals with inter-workgroup load imbalance.

In PredictiveNet [13], Lin et.al proposed a predictive convolutional neural network which can predict the output of convolutional layers and skip over trivial (ineffectual) multiplications. They show that PredictiveNet can decrease the computational cost by a factor of 2.9x for MNIST handwritten digits dataset. The proposed technique is supported by the mean squared error perseverance. Although they reported marginal accuracy loss, they reduced computational cost.

Background

A capsule is a set of neurons that provide vectors as output (instead of scalars in CNN) which represent different properties called the instantiation parameters for a type of entity [12]. The length of these vectors shows the probability of an entity's existence. There are multiple layers of capsules (here two layers). Lower-level capsules send their predictions using transformation matrices to the higher-level capsules. This is done via the routing-by-agreement algorithm. The activation of a higher-level capsule depends on the agreement of predictions. The proposed structure in [12] achieves state-of-the-art accuracy on MNIST dataset. It also has several advantages over conventional CNNs. CapsNet is better than CNN in recognizing highly overlapping datasets. It is also more robust to Affine Transformations applied to the input data.

MNIST dataset of handwritten digits is one of the most popular datasets in machine learning applications. It has 60000 images for training and 10000 images for testing. Each image is 28 by 28 pixels. Each image depicts a digit between 0 and 9. In [12], they also tested other dataset including Fashion MNIST. Fashion MNIST is an image dataset which has the same dataset size, number of classes and image size as MNIST dataset, but it contains images of different types of clothing. Fashion-MNIST serves as a replacement for the original MNIST dataset for benchmarking.

As presented in Figure 1, the architecture of the CapsNet includes two convolutional layers. The output of the second convolutional layer is reshaped to 8-d vectors which are called Primary Capsules. Next, a weight matrix is multiplied by these vectors to create a new set of vectors. Our analysis on skipping zeros is performed on these vectors. The next layer consists of 10 vectors each with 16 dimensions (DigitCaps). Here is where the Dynamic-Routing algorithm comes in. The algorithm aims at finding an agreement between the DigitCaps and the vectors on the previous layer.

The work in [15] includes investigations on pruning. Pruning relies on setting a threshold on values of neurons. In [15] the threshold values are determined for each layer of network separately. For simplicity, they have just used powers of two as threshold values. Any value below the threshold will be truncated to zero. CNVLUTIN architecture performs pruning dynamically. In CNVLUTIN, pruning has no effects on accuracy up to a certain threshold. After that, accuracy starts to decline. However, since many values are set to zero and consequently skipped, overall, we gain in speedup.

In this work we analyze the values of the vectors during the training process. For the training process, dataset is divided to several splits: training, validation and testing sets. Validation set is used to verify the progress of the network. Our analysis is done on the validation set. To optimize a neural network, a dataset is divided to fixed-size batches and these batches are fed to an optimizer. The procedure of applying the whole dataset to the neural network is called an epoch. We store all the values over different iterations in each epoch and inspect them to show what percentage of data consists of zeros.

3 Experiments and Results

We analyze the values of the predictions made by the primary capsules while training the CapsNet on two different datasets i.e. MNIST and Fashion-MNIST. The predictions of the primary capsules are in fact the output of the secondary feature map reshaped into 8-dimensional vectors and then multiplied by the weight matrix. We save all the tensor values over all iterations of validation and then analyze and investigate the results from different perspectives. There is a total of TxBx1152x10x16 numbers, where T is number of iterations per epoch and B is batch size. B is 64 for both datasets and T is 78 and 156 for MNIST and F-MNIST respectively. This leads to a total of 920,125,440 and 1,840,250,880 values per epoch for MNIST and F-MNIST respectively.

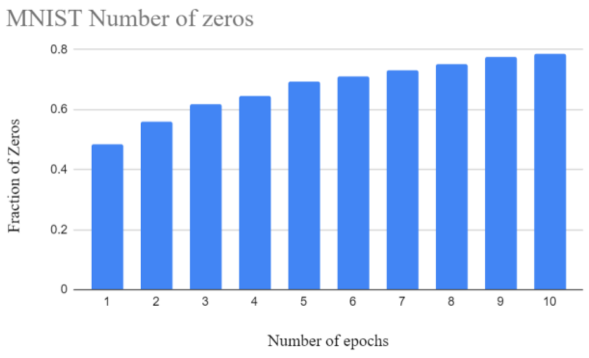


Figure 2. Fraction of zeros over epochs for MNIST

The range of tensor values is within [-1.2312, 1.3018] and [-1.2766, 1.1403] for MNIST and F-MNIST respectively. Both datasets have a significant number of zeros during the training process. Our first observation is that the number of zeros increases as the training makes progress. Figures 2 and 3 show the fraction of zero values over different epochs for both datasets. As the accuracy increases, features get more specific. When the accuracy of the model improves, the number of zeros increases. This is because during checking for specific features in images, many of them are found to be absent. This is the result of the fact that each feature exists in only a different set of images. For the very same reason, F-MNIST has fewer number of absolute zeros over all epochs compared to MNIST. This can be justified by the higher accuracy the network achieves while trained on MNIST (99.42%) compared to F-MNIST (90.45%).

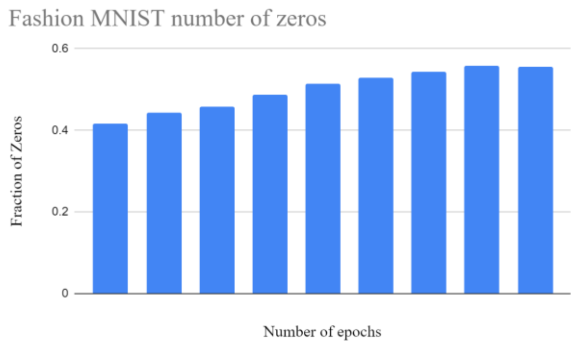
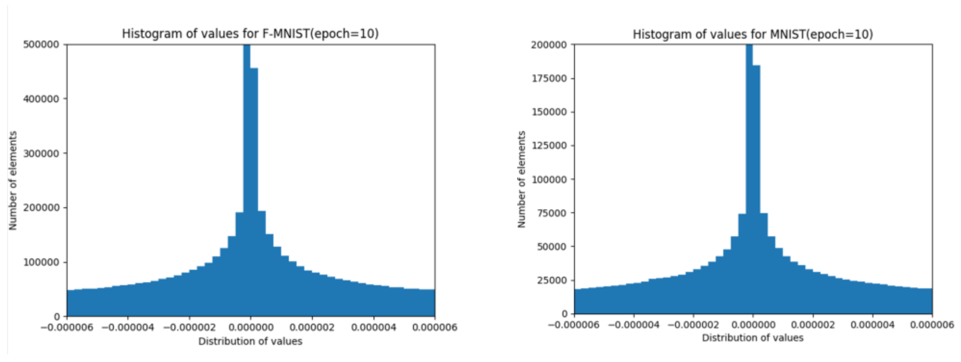


Figure 3. Fraction of zeros over epochs for F-MNIST

Both datasets get high number of absolute zero values. This percentage is ranging from 32% to 77% in the first and 10th epoch for MNIST and from 42% to 55% for FMNIST.

Figure 4(a) and 4(b) depicts histograms of the distribution of predictions generated by the primary capsules for the datasets. Both plots are cropped from the top to make the surrounding points visible. In other words, the bin regarding the number of zeros is removed since it has a significantly higher number of values than other bins. Even though there are many values close to zero, their number is far less that the number of absolute zeros available. As a result, the reduction in computations for values near to zero would be negligible compared to that of absolute zeros. Therefore, for our case pruning does not apply.

Our investigation shows that more than 50% of primary capsule predictions consist of absolute zeros. Multiplications with these zero values can be skipped using methods like the one proposed in CNVLUTIN paper [15] and this would result in fewer computations.



(a) **(b)**

Figure 4. Distribution of predictions of PC for FMNIST (a) and MNIST (b) datasets

4 Conclusion

CapsNet is considered as the next generation neural networks since it not only achieves state-of-the-art results on MNIST handwritten digits dataset, but also shows additional advantages over the conventional CNNs e.g., detecting overlapping entities. CapsNet is computationally expensive. In this work, we investigated the potential of zero skipping in Capsnet and showed that CapsNet has enough zeros to be skipped when trained on MNIST and Fashion-MNIST. This work shows that there is significant room in saving energy and timing in Capsule Networks by using a combination of skipping trivial operation solutions.

Acknowledgements

This research has been funded in part or completely by the Computing Hardware for Emerging Intelligent Sensory Applications (COHESA) project. COHESA is financed under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP485577-15."

References

[1] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, 2017.

[2] J. B. Pendry *et al.*, "References and Notes Supporting Online Material Reducing the Dimensionality of Data with Neural Networks," *IEEE Trans. Microw. Theory Tech*, vol. 47, no. 9, p. 653, 1999.

[3] J. Zhang, L. Ma, and Y. Liu, "Passivity analysis for discrete-time neural networks with mixed time-delays and randomly occurring quantization effects," *Neurocomputing*, 2016.

[4] F. Yang, H. Dong, Z. Wang, W. Ren, and F. E. Alsaadi, "A new approach to non-fragile state estimation for continuous neural networks with time-delays," *Neurocomputing*, 2016.

[5] Y. Yu, H. Dong, Z. Wang, W. Ren, and F. E. Alsaadi, "Design of non-fragile state estimators for discrete time-delayed neural networks with parameter uncertainties," *Neurocomputing*, 2016.

[6] N. Hou, H. Dong, Z. Wang, W. Ren, and F. E. Alsaadi, "Non-fragile state estimation for discrete Markovian jumping neural networks," *Neurocomputing*, 2016.

[7] C. Xiang, L. Zhang, Y. Tang, W. Zou, and C. Xu, "MS-CapsNet: A Novel Multi-Scale Capsule Network," *IEEE Signal Process. Lett.*, vol. 25, no. 12, pp. 1850–1854, 2018.

[8] J. Rajasegaran, V. Jayasundara, S. Jayasekara, H. Jayasekara, S. Seneviratne, and R. Rodrigo, "DeepCaps: Going Deeper with Capsule Networks," 2019.

[9] A. Mobiny and H. Van Nguyen, "Fast CapsNet for lung cancer screening," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 11071 LNCS, pp. 741–749.

[10] Y. Kim, P. Wang, Y. Zhu, and L. Mihaylova, "A Capsule Network for Traffic Speed Prediction in Complex Road Networks," in *2018 Symposium on Sensor Data Fusion: Trends, Solutions, Applications, SDF 2018*, 2018.

[11] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 6791 LNCS, no. PART 1, pp. 44–51.

[12] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic Routing Between Capsules,” no. Nips, 2017.
[13] Y. Lin, C. Sakr, Y. Kim, and N. Shanbhag, “PredictiveNet: An energy-efficient convolutional neural network via zero prediction,” in *Proceedings - IEEE International Symposium on Circuits and Systems*, 2017.
[14] D. Kim, J. Ahn, and S. Yoo, “ZeNA: Zero-Aware Neural Network Accelerator,” *IEEE Des. Test*, vol. 35, no. 1, pp. 39–46, Feb. 2018.
[15] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. Enright Jerger, and A. Moshovos, “Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing,” 2016.
[16] LECUN and Y., “THE MNIST DATABASE of handwritten digits,” *http://yann.lecun.com/exdb/mnist/*.
[17] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms,” Aug. 2017.
[18] T. Luo *et al.*, “DaDianNao: A neural network supercomputer,” *IEEE Trans. Comput.*, vol. 66, no. 1, pp. 73–88, Jan. 2017.