Understanding the Limitations of Existing Energy-Efficient Design Approaches for Deep Neural Networks

Yu-Hsin Chen
Tien-Ju Yang
Massachusetts Institute of Technology
Cambridge, MA
yhchen@mit.edu,tjy@mit.edu

Joel Emer
NVIDIA/Massachusetts Institute of Technology
Westford, MA
jsemer@mit.edu

Vivienne Sze
Massachusetts Institute of Technology
Cambridge, MA
sze@mit.edu

ABSTRACT

Deep neural networks (DNNs) are currently widely used for many artificial intelligence (AI) applications including computer vision, speech recognition, and robotics. While DNNs deliver state-of-the-art accuracy on many AI tasks, it comes at the cost of high computational complexity. Accordingly, there has been a significant amount of research on the topic of energy-efficient processing of DNNs, from the design of efficient DNN algorithms to the design of efficient DNN processors. However, in surveying these techniques, we found that there were certain limitations to the approaches used in this large body of work that need to be addressed. First, the number of weights and MACs are not sufficient for evaluating the energy consumption of DNNs; rather than focusing of weights and MACs, designers of efficient DNN algorithms should more directly target energy and incorporate that into their design. Second, the wide range techniques used for efficient DNN algorithm design has resulted in a more diverse set of DNNs, and the DNN hardware used to process these DNNs should be sufficiently flexible to support these techniques efficiently. Many of the existing DNN processors rely on certain properties of the DNN which cannot be guaranteed (e.g., fixed weight sparsity, large number of channels, large batch size). In this work, we highlight recent and ongoing work that aim to address these limitations, namely energy-aware pruning, and a flexible accelerator (Eyeriss v2) that is computationally efficient across a wide range of diverse DNNs.

ACM Reference Format:
Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. 2018. Understanding the Limitations of Existing Energy-Efficient Design Approaches for Deep Neural Networks. In *Proceedings of SysML Conference (SYSML'18).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Deep neural networks (DNNs) have demonstrated state-of-the-art performance on many artificial intelligence (AI) applications, such as computer vision, speech recognition, and robotics. However, DNN-based methods require significantly more computation than traditional methods, which leads to high energy consumption. For instance, object detection using DNNs requires 311× to 13,486× more energy than traditional histogram of oriented gradients (HOG) features as discussed in [15]. This not only increases the operating cost of data centers, but also is an impediment to deploying DNNs on mobile devices, where the energy budget is limited. Local processing on mobile devices is becoming increasingly preferred due to privacy/security concerns and latency requirements. Designing energy-efficient DNNs and DNN processors is thus critical to realizing mobile AI applications.

In recent years, researchers have approached this objective from both the algorithm and hardware architecture perspective. When summarizing this large body of work for our recent tutorial paper [16], we noticed that there were significant limitations to existing approaches that needed to be addressed. In this work, we will highlight these limitations and summarize our recent and ongoing work that aims to address them.

SYSML'18, February 2018, Stanford, California USA
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 ENERGY-AWARE ALGORITHM DESIGN

From the algorithm perspective, the focus of recent work has mainly been on reducing the number of weights and operations, specifically multiplies and accumulates (MACs), as well as reducing bit-width. Popular techniques to reduce the number of weights and operations include pruning [5] to increase sparsity in weights, and compact network architectures by changing filter shapes using filter decomposition (e.g., SqueezeNet [8], MobileNet [7]).

While these approaches reduce the memory footprint, they do not necessarily translate into a reduction in energy consumption or deliver energy reduction that is proportional to the weight or MAC reduction. For instance, while SqueezeNet requires 50× fewer weights than AlexNet [10], it consumes more energy than AlexNet on a variety of platforms [11, 17]. Another example is that pruning is most effective on the fully connected (FC) layers [5]; however, the energy consumption of a DNN is often dominated by the convolutional (CONV) layers since they require significantly more feature map data movement [2].

There are two main reasons why the number of weights and MACs do not necessarily map to energy consumption: 1) data movement, rather than computation, dominates energy consumption, and 2) the memory hierarchy and dataflow have a large impact on the energy consumption of data movement. The energy consumption of the weights (along with other required data) depends on their movement through the memory hierarchy, which can vary significantly for different weights. Furthermore, the data movement of feature maps needs to be taken into account. As a result, the number of weights is also not a good approximation for energy.

To address the limitations in efficient algorithm design, we will present an energy-aware design approach that directly targets the energy consumption of DNNs. It uses an online DNN energy estimation tool (https://energyestimation.mit.edu/) to enable fast and easy DNN energy estimation. In addition to providing insights for DNN design, this tool can be incorporated into the optimization loop of techniques that aim to reduce energy consumption. This is

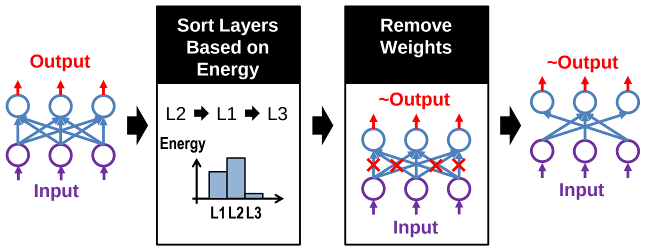

Figure 1: The algorithm flow of energy-aware pruning [17].

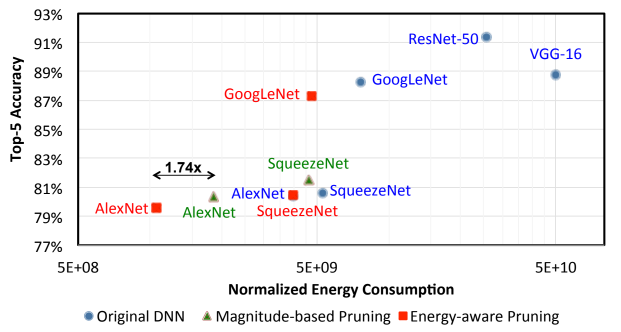

Figure 2: The accuracy-energy trade-off of DNNs pruned by energy-aware pruning and magnitude-based pruning [5].

demonstrated in an approach called energy-aware pruning (EAP) (Fig. 1), which was recently presented at CVPR 2017 [17].

EAP uses the energy to guide a layer-by-layer pruning algorithm. A layer-by-layer pruning algorithm picks one layer, prunes it, fixes it and moves on to prune the next layer. As more layers are pruned, pruning gets more difficult. Accordingly, the layers pruned early on tend to have more weights removed. Thus, to achieve higher energy reduction, EAP prunes the layers that consume most of the energy first. In addition, EAP prunes the weights that have the smallest joint influence on the output feature map rather than the weights with the smallest magnitude. By taking weight correlation into account, EAP is able to prune more weights with the same accuracy compared to existing magnitude-based pruning approaches [5].

EAP reduces the overall energy across all layers by 3.7× for AlexNet, which is 1.74× more efficient than magnitude-based approaches [5] as shown in Fig. 2. Since it is well known that AlexNet is over-parameterized, EAP was also applied to GoogleNet, which is already a small DNN model, to show that it can achieve a 1.6× energy reduction.

3 FLEXIBLE ENERGY-EFFICIENT HARDWARE (EYERISS V2)

From the hardware architecture perspective, the focus of recent work has mainly been on reducing the energy overhead for delivering the data to the MAC engines, as it is well known that data movement consumes significantly more energy than computation [6]. *While existing specialized DNN hardware (ASIC, FPGA) provide improved energy-efficiency compared to GPUs, they often come at the cost of flexibility in terms of the types of layers (CONV or FC) and the type of network architectures/filter shapes that it can support; specifically, they rely on certain properties of the DNN in order to achieve high energy-efficiency.* However, the wide range techniques used for efficient DNN algorithm design has resulted in a more diverse set of DNNs, and there is no guarantee that the algorithm designer will use a certain network architecture or efficient design approach. Thus, DNN hardware needs to be flexible enough to be efficient across all these possible combinations (dense/sparse, number of channels, batch size).

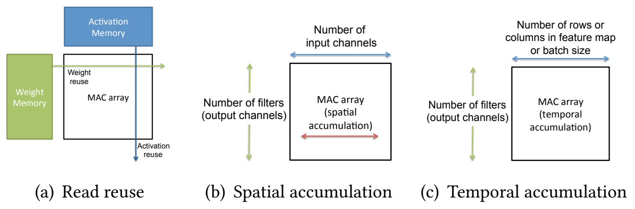
Activation Memory
Weight Memory
Weight reuse
MAC array
Activation reuse
Number of input channels
Number of filters (output channels)
MAC array (spatial accumulation)
Number of rows or columns in feature map or batch size
Number of filters (output channels)
MAC array (temporal accumulation)
(a) Read reuse
(b) Spatial accumulation
(c) Temporal accumulation

Figure 3: While the MAC array provides data reuse of the reads from memory, the amount of reuse and utilization depends on the filter shape, feature map and batch size

Many of the existing specialized hardware in both academia [1, 12, 18] and industry [9, 13] use an architecture consisting of a large MAC array that can exploit reuse of the weights and activations as shown in Fig. 3(a). The degree of reuse and array utilization depends on the number of output channels, and either the number of input channels (Fig. 3(b)) or the feature map/batch size (Fig. 3(c)). While this might be effective for traditional large DNNs such as AlexNet that have many output channels, recent compact networks such as MobileNet exploit group convolutions with fewer channels per group. Furthermore, large batch sizes pose a problem for low latency applications. Finally, it is difficult for these architectures to exploit sparsity due to pruning.

Specialized sparse DNN processors, such as SCNN [14] and EIE [4], have been designed to translate the sparsity from pruning into improved energy efficiency by performing only the non-zero MAC operations and moving/storing the data in compressed form. While this is efficient for sparse DNNs, there would be significant overhead for processing dense DNNs in the compressed format. This is a challenge, since there is no guarantee of sparsity in the DNN. Furthermore, SCNN is optimized for CONV layers while EIE is optimized for FC layers.

To address this, we will present the next generation of our Eyeriss accelerator work [3], a flexible energy-efficient accelerator that can efficiently support DNNs with both sparse and dense weights and/or activations, along with the wider variety of filter shapes that result from filter decomposition used for compact network architectures. This is ongoing work targeted for publication in 2018.

REFERENCES

[1] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. 2014. DianNao: A Small-footprint High-throughput Accelerator for Ubiquitous Machine-learning. In *ASPLOS*.
[2] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. 2016. Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks. In *ISCA*.
[3] Yu-Hsin Chen, Tushar Krishna, Joel Emer, and Vivienne Sze. 2016. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. In *ISSCC*.
[4] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. 2016. EIE: efficient inference engine on compressed deep neural network. In *ISCA*.

[5] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both Weights and Connections for Efficient Neural Network. In *NIPS*.

[6] Mark Horowitz. 2014. Computing's energy problem (and what we can do about it). In *ISSCC*.

[7] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[8] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. 2017. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *ICLR* (2017).

[9] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ACM, 1–12.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*.

[11] D. Moloney. 2016. Embedded deep neural networks: x201C;The cost of everything and the value of nothing x201D;. In *2016 IEEE Hot Chips 28 Symposium (HCS)*. 1–20. https://doi.org/10.1109/HOTCHIPS.2016.7936219

[12] Bert Moons and Marian Verhelst. 2016. A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets. In *Symp. on VLSI*.

[13] Nvidia. 2017. NVDLA Open Source Project. (2017). http://nvdla.org/

[14] Angshuman Parashar, Minsoo Rhu, Anurag Mukkara, Antonio Puglielli, Rangharajan Venkatesan, Brucek Khailany, Joel Emer, Stephen W Keckler, and William J Dally. 2017. SCNN: An Accelerator for Compressed-sparse Convolutional Neural Networks. In *ISCA*.

[15] A. Suleiman, Y. H. Chen, J. Emer, and V. Sze. 2017. Towards closing the energy gap between HOG and CNN features for embedded vision. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–4. https://doi.org/10.1109/ISCAS.2017.8050341

[16] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer. 2017. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proc. IEEE* 105, 12 (Dec 2017), 2295–2329. https://doi.org/10.1109/JPROC.2017.2761740

[17] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. 2017. Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning. In *CVPR*.

[18] Shouyi Yin, Peng Ouyang, Shibin Tang, Fengbin Tu, Xiudong Li, Leibo Liu, and Shaojun Wei. 2017. A 1.06-to-5.09 TOPS/W reconfigurable hybrid-neural-network processor for deep learning applications. In *VLSI Circuits, 2017 Symposium on*. IEEE, C26–C27.