Not All Ops Are Created Equal!

Liangzhen Lai
Arm Inc.
liangzhen.lai@arm.com

Naveen Suda
Arm Inc.
naveen.suda@arm.com

Vikas Chandra
Arm Inc.
vikas.chandra@arm.com

ABSTRACT

Efficient and compact neural network models are essential for enabling the deployment on mobile and embedded devices. In this work, we point out that typical design metrics for gauging the efficiency of neural network architectures – total number of operations and parameters – are not sufficient. These metrics may not accurately correlate with the actual deployment metrics such as energy and memory footprint. We show that throughput and energy varies by up to 5X across different neural network operation types on an off-the-shelf Arm Cortex-M7 microcontroller. Furthermore, we show that the memory required for activation data also need to be considered, apart from the model parameters, for network architecture exploration studies.

1 INTRODUCTION

Exploring efficient neural network (NN) architectures targeted for mobile and embedded devices with constrained energy and memory resources has been the recent trend in machine learning research [3, 5, 9, 12, 13]. Most research use the number of operations (Ops) and/or parameters (i.e., weights) as the metrics for evaluating the model complexity and compactness. While these metrics are sufficient when comparing significantly different NN models (e.g. AlexNet [7] vs. MobileNets [3]), they may not be accurate enough for comparing networks whose complexity and sizes are similar. Furthermore, as research shifts towards fine-grained optimization, e.g., network architecture search [2, 14] and hyperparameter search [10, 13], reductions in Ops or parameters may not always improve the network efficiency.

Energy per inference and total memory footprint are two main system metrics to be considered for deploying NN based solutions on resource constrained devices. In this work, we show examples that NNs with similar network design metrics can have very different deployment metrics when running on resource constrained devices like microcontrollers. In particular, we show:

- Throughput and energy efficiency for different types of NN operations can vary by up to 5X. This can result in 30% difference in runtime and energy for NNs with similar Ops and accuracy.
- Different operations with same amount of weights can have varying amount of activation data, and thus different memory footprint. This may not be an issue for large-scale systems, but is critical for devices with limited memory.

All experiments are performed using optimized neural network kernels in CMSIS-NN [8]. The delay/power results are measured on a NUCLEO-F746ZG mbed development board [1], which has an Arm Cortex-M7 core (running at 216 MHz), 1 MB flash and 320 KB SRAM.

2 ENERGY PER INFERENCE

Energy consumption per inference is a crucial metric that determines the battery life of an embedded system and it is imperative that NN models are optimized for energy efficiency. Typically, number of Ops is considered as a proxy for the energy consumption
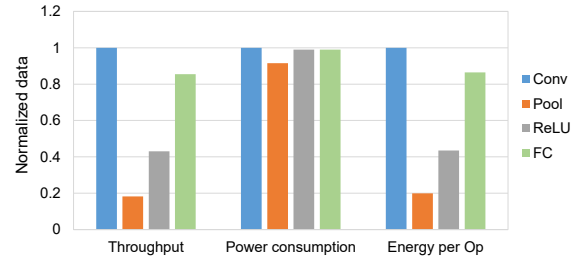

Figure 1: Normalized throughput and energy of different type of NN operations in a CNN for CIFAR-10 dataset.

per inference, but the type of operations also has huge impact on the energy. For example, Fig. 1 shows the normalized throughput, power consumption and energy per Op of different NN operation types of the convolutional neural network (CNN) for CIFAR-10 dataset from Caffe examples [6]. The results show that throughput (i.e. Ops/s) can vary by 5X across different operation types, but average power consumption remains almost same. This implies that the overall energy consumption depends mostly on the throughput. Among all the operation types, max pooling is particularly slow because it is based on comparisons (i.e. branch) rather than computations. However, in a typical NN, convolution and fully-connected (FC) layers constitute more than 90% of the operations. These layers achieve good throughput by effectively utilizing the SIMD Multiply-Accumulate (MAC) instructions.

Fig. 2 shows the throughput of different MAC based NN operations. Since the throughput depends heavily on the layer dimensions, we use the number of MAC operations per output to represent the effectiveness of SIMD MAC instructions. In this case, the difference between operation types represents the relative overhead of fetching the MAC operands. In general, convolution is slower than fully-connected layer because of additional *im2col* overhead. However, 1x1 convolution does not require *im2col*. It uses matrix-matrix
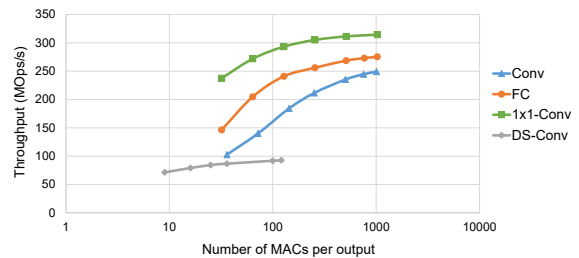

Figure 2: Throughput variation with number of MACs per output for different types of NN operations. Depthwise separable convolution (DS-Conv) typically has much less MACs per output compared to other operation types.
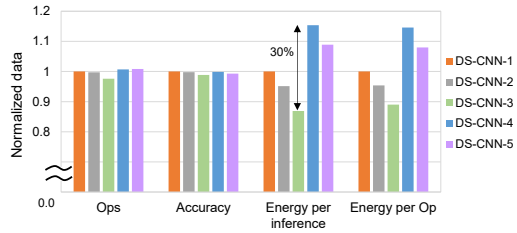
Figure 3: Normalized energy consumption of 5 DS-CNN models with similar accuracy and number of Ops.

multiplication (GEMM) style of computations, which is faster than matrix-vector multiplication (GEMV) style of computations used in fully-connected layer due to better data reuse. Among all operation types, depthwise separable convolution (DS-Conv) is the slowest as it has higher *im2col* overhead and typically lower MACs per output.

Understanding the throughput differences between operation types is crucial for designing efficient NN architectures. Fig. 3 shows the normalized energy consumption, number of Ops and accuracy of 5 DS-CNN models [13] with different number of layers and features per layer, trained on Google speech commands dataset [11]. It shows that the energy per inference varies by as much as 30% across these models although they come from the same NN architecture family and have similar accuracy and total number of Ops.

The distributions of the different operation types of DS-CNN-3 and DS-CNN-4 models are shown in Fig. 4. Compared to the DS-CNN-3 model, DS-CNN-4 has higher proportion of DS-Conv Ops, which has substantially lower throughput compared to other operation types as shown in Fig. 2. This results in 30% reduction in the overall throughput and hence energy efficiency. Using Ops as a metric without considering the throughput of different operation types on the actual hardware may lead to sub-optimal efficiency.
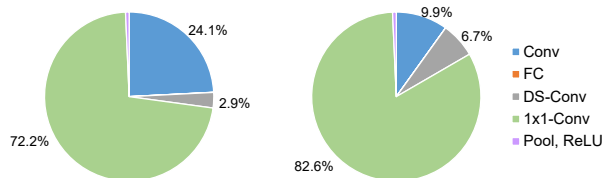

Figure 4: Distribution of different operation types in DS-CNN-3 (left) and DS-CNN-4 (right) models shown in Fig. 3.

When performing fine-grained NN optimization, the operation type and dimensions should be considered for evaluating the network efficiency. The results we show in this work are based on a general purpose processor and the operation characteristics for other platforms (e.g. GPU, FPGA, DSP, accelerator) can be very different. Performance for different operation types, similar to results in Fig. 2, can be pre-characterized for the target hardware platform and used for estimating the network efficiency.

3 MEMORY FOOTPRINT

System memory size is the other important limiting factor for running NNs on resource constrained devices. For example, typical microcontroller SoC have 100 KB - 1 MB of flash (to store program binary and model weights) and 10-300 KB of SRAM (to store the activation data).

The number of model parameters, which can be used as metric to quantify the compactness of a NN model, determines whether the model fits in the flash or not. However, it may not be a good metric for representing the total memory footprint, as it does not consider the activation data typically stored in the SRAM. The amount of activation data can be a significant part of the total memory footprint and will depend on the operation type as well. For example, Fig. 5 shows the memory footprint of four NN models for the keyword spotting application from [13]. The size of maximum concurrent activation data varies between 1% to 30% of the total memory footprint.
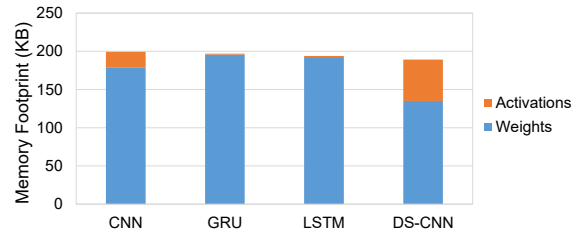

Figure 5: Memory footprint (total weights and maximum concurrent activation data) breakdown for four different types of models from [13].

Apart from operation types, NN topology can also affect the size of maximum concurrent activation data. Regular feed-forward network need to store only the input and output data of the current layer. If there are other feed-forward connections, such as in DenseNet [4], the total number of concurrent activation data will increase. Also, some networks generated by automatic network architecture search can have many feed-forward connections [14], which can substantially (~10X) increase the size of the activation data.

The network optimization target in NN architecture exploration should be the total memory footprint instead of the number of parameters. During NN architecture exploration, total memory footprint can be estimated by the sum of the size of maximum concurrent activation data and the size of weight parameters. The maximum concurrent activation data can be obtained from the network graph and the execution order. The concurrent activation data, when executing an operation, will include the operation input, output, as well as other activation data (i.e. feed-forward edges) that are needed for the operation.

4 CONCLUSION

In this work, we show that the NN operation type has significant impact on system efficiency. The commonly used network design metrics – number of operations and parameters – need to be rethought as they may not accurately correlate with system design metrics such as energy efficiency and memory footprint. Experimental results on an off-the-shelf Arm Cortex-M microcontroller show that the energy per operation can vary up to 5X for different NN operation types. Network activation data, which are typically overlooked can also contribute to up to 30% of the total memory footprint. The network architecture exploration should account for both energy efficiency as well as total memory footprint to make the inference more efficient on resource constrained devices.

REFERENCES

[1] NUCLEO-F746ZG development board. http://www.st.com/en/evaluation-tools/nucleo-f746zg.html.

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. 2016. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167* (2016).

[3] 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).

[4] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2016. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993* (2016).

[5] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:1602.07360* (2016).

[6] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 675–678.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[8] Liangzhen Lai, Naveen Suda, and Vikas Chandra. 2018. CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs. *arXiv preprint arXiv:1801.06601* (2018).

[9] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*. Springer, 525–542.

[10] Dimitrios Stamoulis, Ermao Cai, Da-Cheng Juan, and Diana Marculescu. 2017. HyperPower: Power-and Memory-Constrained Hyper-Parameter Optimization for Neural Networks. *arXiv preprint arXiv:1712.02446* (2017).

[11] Pete Warden. 2017. Speech Commands: A public dataset for single-word speech recognition. *Dataset available from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz* (2017).

[12] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2017. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083* (2017).

[13] Yundong Zhang, Naveen Suda, Liangzhen Lai, and Vikas Chandra. 2017. Hello Edge: Keyword Spotting on Microcontrollers. *arXiv preprint arXiv:1711.07128* (2017).

[14] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).