A method to estimate the energy consumption of deep neural networks

A Method to Estimate the Energy Consumption of Deep Neural Networks

Tien-Ju Yang, Yu-Hsin Chen, Joel Emer, Vivienne Sze
Massachusetts Institute of Technology, Cambridge, MA, USA
{tjy, yhchen, jsemer, sze}@mit.edu

***Abstract*—Deep Neural Networks (DNNs) have enabled state-of-the-art accuracy on many challenging artificial intelligence tasks. While most of the computation currently resides in the cloud, it is desirable to embed DNN processing locally near the sensor due to privacy, security, and latency concerns or limitations in communication bandwidth. Accordingly, there has been increasing interest in the research community to design energy-efficient DNNs. However, estimating energy consumption from the DNN model is much more difficult than other metrics such as storage cost (model size) and throughput (number of operations). This is due to the fact that a significant portion of the energy is consumed by data movement, which is difficult to extract directly from the DNN model. This work proposes an energy estimation methodology that can estimate the energy consumption of a DNN based on its architecture, sparsity, and bitwidth. This methodology can be used to evaluate the various DNN architectures and energy-efficient techniques that are currently being proposed in the field and guide the design of energy-efficient DNNs. We have released an online version of the energy estimation tool at *energyestimation.mit.edu*. We believe that this method will play a critical role in bridging the gap between algorithm and hardware design and provide useful insights for the development of energy-efficient DNNs.**

***Index Terms*—Deep learning, deep neural network, energy estimation, energy metric, machine learning.**

I. Introduction

Deep neural networks (DNNs) have demonstrated state-of-the-art performance on many artificial intelligence (AI) applications, such as computer vision, speech recognition, machine translation, etc. However, DNN-based methods require significantly more computation than the traditional methods, which leads to high energy consumption. This not only increases the operating cost of data centers, but also prevents DNNs from being deployed 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 is critical to realizing mobile AI applications.

To close the gap between DNN design and energy optimization, we propose an energy estimation methodology for DNNs in this paper. The proposed methodology will help DNN designers understand the various design trade-offs and enable them to use this knowledge to guide the design of energy-efficient DNNs. For example, when designing a DNN, several hyperparameters (e.g., number of layers, number of filters in each layer, width and height of the filters, etc.) need to be determined. These hyperparameters will have a profound impact on the DNN's accuracy and energy consumption. With this methodology, the designer is able to analyze the impact of each hyperparameter on energy to make a better decision. This leads to a more energy-efficient DNN and hence enables more DNN-based mobile applications. To enable fast and simple DNN energy estimation, we made the energy estimation tool available online at https://energyestimation.mit.edu/.

Although this paper focuses on deep convolutional neural networks (DCNNs), the concepts discussed and the overall conclusions also apply to other types of DNNs. DCNNs are composed of several types of layers, such as convolution (CONV), non-linearity, normalization, pooling, etc. Since the CONV layers dominate the overall computation and energy consumption, the proposed methodology focuses on estimating the energy consumption of CONV layers as well as fully-connected (FC) layers, which can be viewed as a special case of CONV layers.

II. The Necessity of Energy Estimation

A CONV layer takes the input feature maps (ifmaps) and convolves them with a series of filters to generate the output feature maps (ofmaps). Each step of the convolution involves performing element-wise multiplication between the filter and the ifmaps, and accumulating all the element-wise products to compute an activation in the ofmaps. Since the accumulation usually cannot be finished in one step, the accumulation is performed in an iterative manner and the intermediate values are called the partial sums. Therefore, the CONV operation can be decomposed into a large number of multiplication-and-accumulation (MAC) operations, which involve ifmaps, filters, ofmaps and partial sums.

Current research on efficient DNN design mostly focuses on reducing the number of filter weights and/or the number of MACs; however, these metrics do not necessarily map to energy consumption. There are two main reasons: 1) data movement, rather than computation, dominates energy consumption, and 2) the memory hierarchy and dataflow has a large impact on the energy consumption of data movement.

Fig. 1 illustrates how a MAC is carried out on a hardware platform. To perform a MAC, the arithmetic logic unit (ALU) takes a filter weight and an ifmap activation, multiplies them and adds the resulting product to the previous partial sum to generate an updated partial sum. The three input values and the output value are stored in memory. As shown in Fig. 2, accessing a value in the external memory (DRAM in this example) consumes significantly more energy than computing

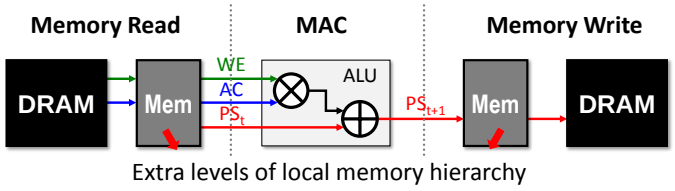

Fig. 1. The memory hierarchy in modern hardware platforms. The arithmetic logic unit (ALU) takes a filter weight (WE) and an input feature map activation (AC), multiplies them and adds it to the previous partial sum (PS$_t$) to generate an updated partial sum (PS$_{t+1}$). The three input values and the output value are stored in the memory hierarchy.

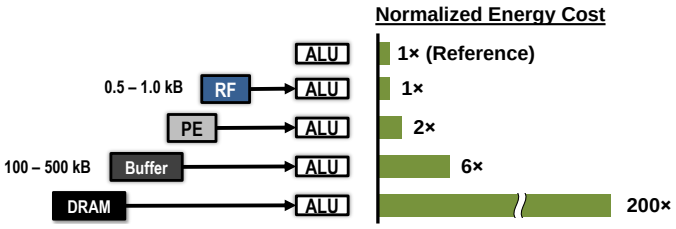

Fig. 2. An example of the energy consumption of memory access and computation on a commercial 65nm process. The unit of energy is normalized in terms of the energy for a MAC operation (i.e., 200 = energy of 200 MACs) [1].

a MAC. Because of this large gap in energy consumption between external memory access and computation, modern hardware systems insert extra levels of local memory on-chip between the ALU and the external memory; this is referred to as the memory hierarchy. According to Fig. 2, the local memory levels consume lower energy than the external memory at the cost of much smaller storage capacity. The benefit of the memory hierarchy is that when a piece of required data is read from the external memory, it can be stored in some local memory levels where it can be reused as many times as possible to amortize the high energy consumption of the external memory access. However, memory access from the local memory still consumes more energy than computation and hence data movement dominates energy consumption. For this reason, the number of MACs is not a good metric for energy. Furthermore, the energy consumption of the weights (along with other required data) depends on their movement through the different memory levels, which can vary significantly for different weights. As a result, the number of weights is also not a good approximation for energy.

In summary, a direct energy metric is necessary for guiding the energy-efficient DNN design. In the next section, we will introduce the proposed energy estimation methodology, which serves this purpose.

III. Energy Estimation Methodology

A. Overview

The proposed energy estimation methodology estimates the energy consumption of each layer individually, so that it can give a layer-wise energy breakdown to help analyze the DNN.

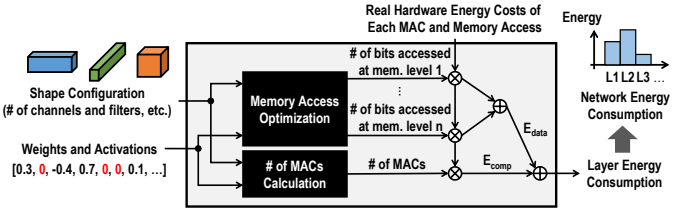

Fig. 3. The proposed energy estimation methodology.

Fig. 3 illustrates the proposed methodology. It takes the shape configuration of a layer (e.g., number of filters, number of ifmaps, width and height of the filters, etc.) and the values and bitwidths of the filter weights and the ifmap activations as inputs and estimates the energy consumption of the layer. The network-wise energy is the summation of the energy of all the layers.

We formulate the layer-wise energy as the following equation:

$$E_{layer} = E_{comp} + E_{data},$$

where E_{comp} is the computation energy (the lower part of Fig. 3), which corresponds to the energy of performing MACs, and E_{data} is the data energy (the upper part of Fig. 3), which corresponds to the energy of data movement. The data energy can be further decomposed into the data energy of the three data types, which are filters, ifmaps and ofmaps. The data energy of the partial sums is included in that of the ofmaps. For estimating the computation energy, we first calculate the number of MACs and scale it by the hardware energy cost of performing a MAC. Therefore, the computation energy only depends on the number of MACs.

The data movement energy is more difficult to estimate than the computation energy. As discussed in Sec. II, modern hardware platforms use a memory hierarchy and how data moves in the hierarchy (i.e., the *dataflow*) significantly influences the energy consumption. Therefore, before estimating the data energy, memory access optimization needs to be carried out to find an efficient dataflow. The number of bits accessed at each memory level is then calculated based on the dataflow and scaled by the hardware energy cost of accessing one bit at that memory level. The data energy is the sum of the energy consumed at all the memory levels and does not only depend on the number of MACs.

B. Memory Access Optimization

The goal of the memory access optimization is to find an efficient dataflow that minimizes the energy consumption under the hardware constraints. Ideally, we would like to put all the data in the local memory to avoid access to the expensive external memory (in terms of energy). However, it is usually infeasible because the storage capacity of the local memory is limited due to its high area cost. As a result, we are only able to hold a *chunk* of data in the local memory at a time and maximize its reuse.

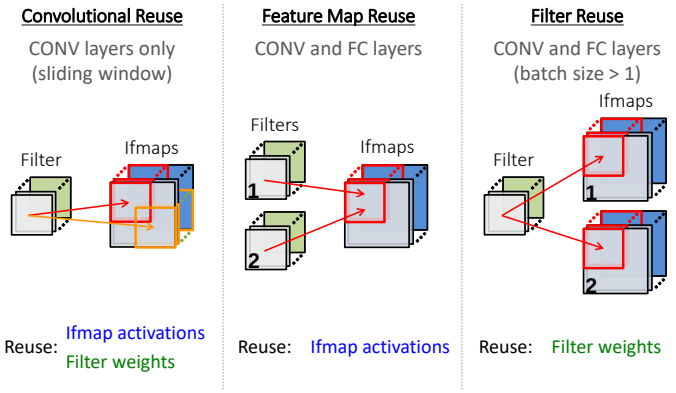

Fig. 4. Three types of data reuse opportunities in CONV and FC layers [1].

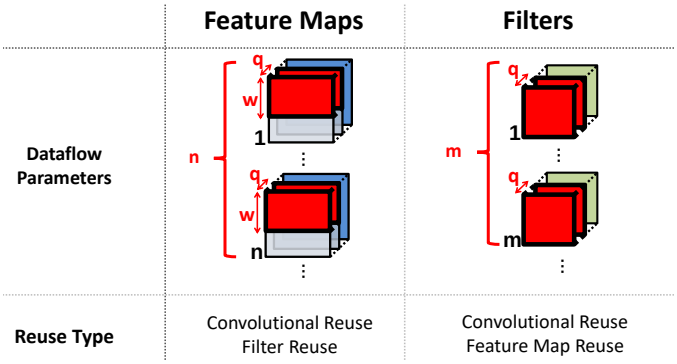

Fig. 5. A few key dataflow parameters of the row-stationary dataflow [2].

The high-dimensional convolution in CONV layers has three types of data reuse opportunities as shown in Fig. 4 [1].

- **Convolutional reuse**: The same filter weights and the same ifmap activations can be reused to generate activations in an ofmap.
- **Feature map reuse**: The weights from different filters can reuse the same ifmap activations to generate the corresponding ofmap.
- **Filter reuse**: The same filter weights can be reused on activations from different ifmaps when multiple images are processed at once (i.e., batch processing).

Unfortunately, there is a trade-off between the three types of data reuse under the hardware constraints. We take the row-stationary dataflow [1], [2] as an example. Fig. 5 illustrates a few key dataflow parameters used by the row-stationary dataflow. The dataflow parameters govern how the data is divided and moves through the memory hierarchy. For instance, each chunk of feature maps contains n sets of q feature maps, and each feature map has only w rows. Each chunk of filters contains m filters, and each filter has only q channels. Generally speaking, we would like to maximize the size of the chunks to increase data reuse, but the chunks of different data types share the same storage space and hence compete with each other. For example, to increase the filter reuse, we can enlarge the chunk of feature maps. However, this decreases the feature map reuse because the size of the filter chunk needs to be reduced due to the fixed storage space. Moreover, the resulting dataflow of the memory access optimization is influenced by the shape configuration of the layer. For instance, if the width and the height of the filters increase, we are forced to reduce other dataflow parameters, such as the size of the feature map chunk, and this reduces the corresponding data reuse. Therefore, to find an energy-efficient dataflow, the memory access optimization solves an optimization problem with the shape configuration as well as the filter weights and the ifmap activations (will be discussed in Sec. III-C) as the inputs. The objective is to minimize the energy consumption under the constraints that the data chunks must be able to fit in each local memory level.

C. Bitwidth and Sparsity

The proposed methodology also accounts for the impact of bitwidths and sparsity. This allows for a more accurate analysis of several well-known energy-efficient techniques, such as network quantization [3] which affects bitwidth, and network pruning [4] which affects sparsity.

- **Bitwidth**: For the computation energy, most of the energy is consumed by the multiplication operations. The multiplication operation takes two inputs: a filter weight and an ifmap activation. Therefore, the computation energy scales linearly with the bitwidth of each input and quadratically with the bitwidths of both inputs. For the data energy, we estimate it based on the number of bits accessed for each of the data types at each memory level. The number of bits accessed scales linearly with the corresponding bitwidth when the dataflow is fixed. However, varying the bitwidths may change the resulting dataflow of the memory access optimization. For example, when the bitwidth of the ifmaps increases, a given memory level may not be able to accommodate the chunk. Thus, we are forced to find another dataflow by performing the memory access optimization.
- **Sparsity**: There are two sources of sparsity. The first source is the sparsity in the feature maps. ReLU non-linearity is widely used in DNNs, especially for computer vision applications. The ReLU non-linearity zeros out the negative activations in the feature maps and generates sparse feature maps. The second source is the sparsity in the filters of pruned DNNs. To make DNN training easier, the DNNs are usually over-parameterized. Therefore, a large number of filter weights in a DNN are redundant. Network pruning identifies and removes these redundant weights from the DNN, which leads to sparse filters. To factor in sparsity in the computation energy, we assume that a multiplication can be skipped if at least one input is zero because the output would be zero. For the data energy, data compression techniques, such as the widely adopted run-length encoding, can be applied to reduce the number of bits accessed. To estimate energy more precisely, the influence of data compression should be considered in the memory access optimization.

D. Computing Memory Access and Number of MACs with a Simulator

Computing the number of bits accessed and MACs becomes more challenging when we factor in sparsity. With sparsity, the number of bits depends on not only the shape configuration but also the locations of non-zero values in the feature maps and the filters. For example, when we use run-length encoding to compress two streams, (0,1,0) and (0,0,1), the former stream will have 2× number of bits of the latter one in the compressed form. As a result, to get the accurate number of bits and MACs, we need to build and run a simulator to determine how the data is divided into chunks and moves in the memory hierarchy, and then apply data compression on each chunk to compute the number of bits accessed. Finally, we must simulate the convolution operations to count the number of non-skipped MACs, which requires the underlying filter weight values and feature map activation values.

IV. Online DNN Energy Estimation Tool

We provide an online DNN energy estimation tool to enable fast and easy DNN energy estimation, which is available at https://energyestimation.mit.edu/. As an example, the online tool applies the proposed methodology on the Eyeriss platform, which is an energy-efficient reconfigurable DNN processor, with the row-stationary dataflow [2].

A. Computing Memory Access and Number of MACs with an Analytical Method

While a simulator gives accurate numbers, it requires long runtimes for large DNNs. Therefore, we developed an analytical method that can be used instead of the simulator described in Sec. III-D to compute the number of bits accessed and MACs and uses pre-computed dataflows. Specifically, the online tool uses methods to quickly approximate the number of bits and MACs that do not depend on the locations of non-zero values.

To compute the number of bits, the tool uses significance map encoding instead of the run-length encoding. Significance map encoding encodes a floating-point zero value into a one-bit zero value and a floating-point non-zero value into a one-bit one value followed by the original floating-point non-zero value. Therefore, the number of bits in the encoded form only depends on the number of non-zero values, irrespective of their locations.

To compute the number of MACs, the tool scales the number of MACs of the dense convolution by the percentage of the non-zero values in the inputs. For instance, if 50% of the ifmap activations and 40% of the filter weights are non-zero, we only need to compute 20% of the total MACs of the dense counterpart. As a result, the number of MACs only depends on the shape configuration and the number of non-zeros values.

Table I compares the estimated energy using the simulator versus the analytical method on AlexNet [5] and GoogLeNet [6]. The result shows that the difference is within 3% and the analytical method can be used as a fast alternative to the simulator.

TABLE I
The comparison of the estimated energy using the simulator versus the analytical method.

DNN	Simulation	Analytical Method	Difference
AlexNet [5]	4G	3.9G	-2.5%
GoogLeNet [6]	7.6G	7.4G	-2.6%

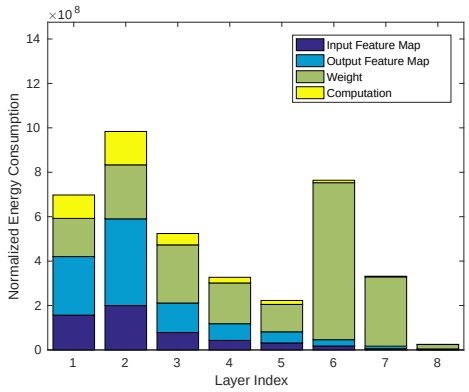

Fig. 6. The estimated energy breakdown of AlexNet [5] generated by the online DNN energy estimation tool at https://energyestimation.mit.edu/.

B. Tool Usage

The tool takes three inputs for each layer: 1) the shape configuration of the layer, 2) the number of non-zero values in the filters and the feature maps and 3) the bitwidths of the filters and the feature maps. The output is a per-layer energy breakdown and we can further decompose the total data energy into the data energy of the three data types. For example, Fig. 6 shows the estimated energy of AlexNet using the online tool.

V. Case Study

A. Understanding DNNs

With the proposed methodology and the tool, we are able to evaluate different DNN architectures and energy-efficient techniques to provide several key observations.

- **Convolutional layers consume more energy than fully-connected layers.** According to Fig. 6, the energy consumption of the CONV layers (layer 1-5, 72%) is higher than that of the FC layers (layer 6-8, 28%) even though the FC layers contain 96% of the total weights. The reason is that the CONV layers involve significantly more feature map data movement, which induces higher feature-map-related energy consumption. We can also observe that the energy consumption of the FC layers is dominated by the weight energy. This is due to the fact that there is no convolutional reuse in the FC layers.
- **Deeper DNNs with fewer weights do not necessarily consume less energy than shallower DNNs with more weights.** To reduce the model size without sacrificing the accuracy, a common strategy is to make the DNN much deeper (i.e., with more layers) and aggressively reduce the size of each layer. However, this strategy does not necessarily lead to lower energy consumption. For example, SqueezeNet [9] is a DNN that has 2.3× number of layers of AlexNet with 51.8× fewer weights.

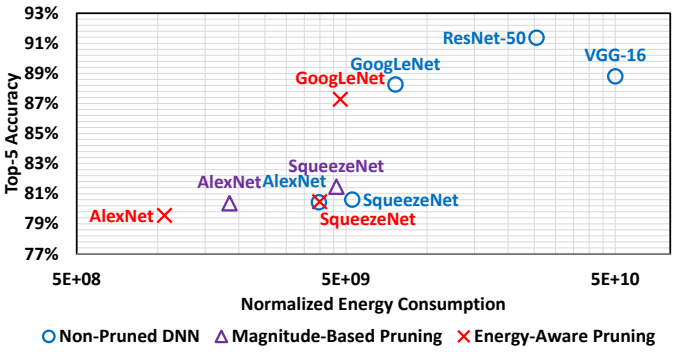

Fig. 7. The accuracy-energy trade-off of well-known DNNs and the corresponding DNNs pruned by energy-aware pruning and magnitude-based pruning [7] on the ImageNet dataset [8]. The estimated energy was generated by using the simulator.

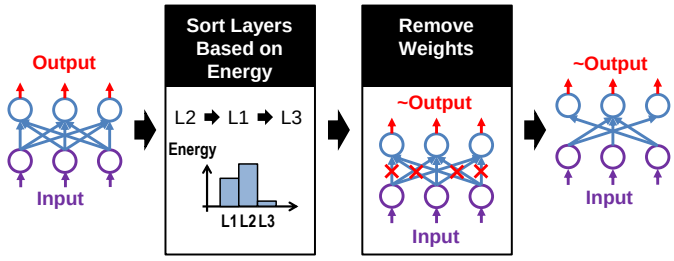

Fig. 8. The simplified algorithm flow of energy-aware pruning [4].

However, it still consumes more energy than AlexNet (Fig. 7). The reason is that when a DNN becomes deeper, it usually generates more feature map data movement and the corresponding energy increase outweighs the energy reduction in moving weights.

- **Data movement is more expensive than computation.** One example is GoogLeNet [6]. In GoogLeNet, the computation energy only accounts for 10% of the total energy. In contrast, moving the feature maps consumes 68% of the total energy.

In summary, data movement dominates the energy consumption and the data movement of feature maps needs to be taken into account, which is usually overlooked.

B. Guiding Energy-Efficient DNN Design

In addition to providing insights for DNN design, the proposed energy estimation methodology can be incorporated into the optimization loop of techniques that aim to reduce energy consumption. This is demonstrated in an approach called energy-aware pruning (EAP) [4] (Fig. 8). EAP improves network pruning primarily in two ways.

First, EAP uses the estimated 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 sorts the layers based on the estimated energy consumption and starts pruning the layers that consume most of the energy first.

Second, EAP prunes the weights having the smallest joint influence on the ofmaps. In contrast, the magnitude-based pruning methods (MBP) [7] remove the weights with the smallest magnitude without considering the correlation between weights. EAP takes weight correlation into account and is able to prune more weights with the same final accuracy.

Fig. 7 illustrates the accuracy-energy trade-off of well-known DNNs and the corresponding DNNs pruned by EAP and MBP [7] on the ImageNet dataset [8]. As we can see, the DNNs pruned by EAP achieve better trade-off than the non-pruned DNNs and the DNNs pruned by MBP. For example, EAP reduces the energy consumption of the non-pruned AlexNet by 3.7× and that of the MBP-pruned AlexNet by 1.7× with similar accuracy. The DNNs pruned by EAP are available at [10].

VI. Conclusion

This work presents an energy estimation methodology for DNNs based on the architecture, bitwidth and sparsity. Along with the methodology, a corresponding online tool is provided at https://energyestimation.mit.edu/. Based on the memory hierarchy in modern hardware platforms and the energy analysis of well-known DNNs, we show that 1) the number of weights and MACs are not good metrics for energy, 2) data movement is more expensive than computation, 3) the data movement of feature maps needs to be taken into account. Moreover, we show an example of how the energy estimation methodology can be used to guide network pruning and achieve better accuracy-energy trade-off. With this methodology and tool, researchers are able to quantify the energy costs of different design choices during design time and hence bridge the gap between DNN algorithm/hardware design and energy optimization.

References

[1] Y. Chen, J. Emer, and V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," in *ISCA*, 2016.

[2] Y. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *JSSC*, 2017.

[3] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," in *ECCV*, 2016.

[4] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning," in *CVPR*, 2017.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *NIPS*, 2012.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper With Convolutions," in *CVPR*, 2015.

[7] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," in *ICLR*, 2016.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[9] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," *arXiv:1602.07360*, 2016.

[10] "Energy-aware pruning website." http://eyeriss.mit.edu/energy.html.