Dynamic bit-width reconfiguration for energy-efficient deep learning hardware

Dynamic Bit-width Reconfiguration for Energy-Efficient Deep Learning Hardware

Daniele Jahier Pagliari
Politecnico di Torino
daniele.jahier@polito.it

Enrico Macii
Politecnico di Torino
enrico.macii@polito.it

Massimo Poncino
Politecnico di Torino
massimo.poncino@polito.it

ABSTRACT

Deep learning models have reached state of the art performance in many machine learning tasks. Benefits in terms of energy, bandwidth, latency, etc., can be obtained by evaluating these models directly within Internet of Things end nodes, rather than in the cloud. This calls for implementations of deep learning tasks that can run in resource limited environments with low energy footprints. Research and industry have recently investigated these aspects, coming up with specialized hardware accelerators for low power deep learning. One effective technique adopted in these devices consists in reducing the bit-width of calculations, exploiting the error resilience of deep learning. However, bit-widths are tipically set statically for a given model, regardless of input data. Unless models are retrained, this solution invariably sacrifices accuracy for energy efficiency.

In this paper, we propose a new approach for implementing input-dependant dynamic bit-width reconfiguration in deep learning accelerators. Our method is based on a fully automatic characterization phase, and can be applied to popular models without retraining. Using the energy data from a real deep learning accelerator chip, we show that 50% energy reduction can be achieved with respect to a static bit-width selection, with less than 1% accuracy loss.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Hardware** → **Application specific integrated circuits**; *Methodologies for EDA*;

KEYWORDS

Energy-efficiency, Deep learning, Energy-quality tradeoff

1 INTRODUCTION

Deep Neural Networks (DNNs) have recently gained popularity thanks to their effectiveness in classification problems in a wide set of domains, such as natural language processing, computer vision, etc [8]. Besides the availability of large datasets, essential for a meaningful learning phase of these networks, another key element in the success of DNNs is the access to high-performance computing resources such as GPUs or large servers [8]. These hardware resources, however, have power requirements in the range of the hundred of Watts, which are clearly only available via direct connection to the power grid.

On the other hand, there is an increasing demand of low-power deep learning hardware that can implement these algorithms in "edge" nodes (mobile, IoT sensors, wearables etc), which besides having limited computational power, are also typically battery-powered [2]. This need applies in particular to the usage of DNNs for *inference*, whereas *training* could still be done in the cloud [6].

One solution to achieve the required power efficiency is to design specialized hardware accelerators for DNNs inference. Many of these accelerators have been already proposed in literature, with particular focus on the class of Convolutional Neural Networks (CNNs) for vision tasks [3, 14–17, 19].

Perhaps the most popular low-power strategy implemented in these designs consists in performing computations at reduced bit-widths, thus trading off accuracy for smaller power consumption [14, 16, 19]. This technique yields benefits both in the data path and in the memory path, thanks to the reduction of the energy cost for data transfers, which usually dominates the total energy consumption for these systems [3].

Bit-width reduction works well due to the well-documented intrinsic error resilience of machine learning tasks. Many works have demonstrated that high-precision computations are often unnecessary in presence of statistical algorithms; moreover, adding noise during training has been shown to improve estimation accuracy [9, 10, 13, 18].

Most of the above works share the characteristic that the bit-width is set *statically*, either for the entire neural network or on a layer-by-layer basis, and is not adapted to different inputs. Static bit-width often causes losses in accuracy, whenever there exist particularly "difficult" inputs that cannot be classified correctly using low precision. While this effect is alleviated by retraining the networks, this means that every time an existing hardware has to be used with a new model: (i) an optimal precision for the model has to be fixed and (ii) the model has to be retrained.

Very few works have addressed the issue of adaptively reconfiguring the precision of DNNs to the characteristics of the inputs. The most effective and closer to the proposed solution, is the method of [15], which applies a "big/little" paradigm to DNNs: "easy" inputs are classified using a "little" DNN, whereas "difficult" ones use a "large" DNN. The limitation of this approach is that it requires a double algorithm design effort, as the "big" and "little" DNNs are effectively two full, separate, hand-crafted networks.

In this work, we propose a new method that is somehow inspired to [15] in its principle, but uses dynamic bit-width adaptation as a mechanism to tune the accuracy and power of a CNN accelerator. Our method does not need retraining and can be applied automatically to an existing CNN model, via a preliminary characterization phase. Moreover, it is hardware agnostic, and can be applied to any CNN accelerator that supports multiple precisions. In this paper, we show its application to the design of [14].

Experiments on two popular CNN models show that our method can reduce inference energy consumption as much as 50% with

respect to a static bit-width solution, with a loss of accuracy of less than 1% with respect to a reference floating point model.

2 BACKGROUND AND RELATED WORK

2.1 Convolutional Neural Networks

Convolutional Neural Network (CNN) models are considered as the state of the art for many vision-related classification tasks [8]. A high level view of a traditional CNN architecture is shown in Figure 1. The network consists of a series of layers, performing feed-forward computations on input *feature maps* to produce output *feature maps*.

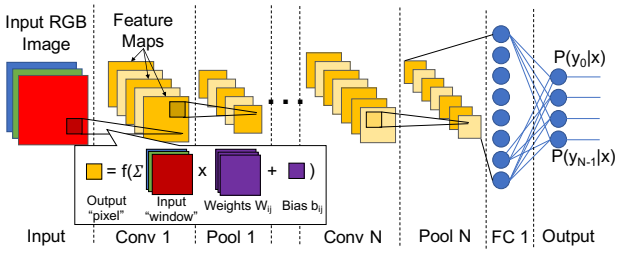


Figure 1: High-level diagram of a CNN.

The first layers are normally an alternation of *convolutional layers* (Conv), which extract local features from the previous layer, and *pooling layers* (Pool), that reduce the dimensionality of feature maps, for model size containment and for improving translation invariance. The last layers constitute the actual classifier and are typically fully connected (FC), i.e. each feature is a function of all features from the previous layer. Many variations on this basic architecture have been devised; for a more extensive review of deep learning and CNN models, readers are referred to [8].

From a computational standpoint, it has been shown that *Conv* layers account for more than 90% of the total number of operations [10]; therefore, these are the key layers to be targeted for reducing energy consumption. The basic function implemented by a Conv layer is shown in the zoomed callout of Figure 1. The j-th output feature map is computed by a filtering operation over a sliding window of elements from all input maps. Elements in the window are multiplied with a matrix of *weights*, and then summed with each other and with a *bias*. A non-linear function (typically a REcitified Linear Unit or ReLU) is applied to the outputs of this summation to obtain one element of the output map. Therefore, the basic operation performed in a Conv layer is a multiply and accumulate (MAC). Values for weights and biases are learned during the *training phase*. Subsequently, the network can be used to classify images in the so called *inference phase*.

2.2 Low-Power CNN Hardware Accelerators

Several works have proposed implementations of low-power HW accelerators for CNNs on FPGAs [20] and ASICs [4, 7, 14, 16, 21]. Many of them rely on some form of computational *approximation* [2], leveraging the fact that machine learning is a typical error tolerant application domain. We will specifically survey these approaches as they are more closely related to our work. Some strategies enable approximations in terms of allowing errors to occur either by simplifying operations in "critical" neurons (e.g., [16, 21]), or by aggressive use of voltage (and/or frequency) scaling on memories and datapath operators [14, 16]. A vast category of solutions relies on *reduced precision* computations, often referred to as network *quantization*. Reduced precision (i.e., bit-width) permits energy savings both in the datapath hardware and in the memory hierarchy, which is often the energy bottleneck, given that state-of-the-art models may have million of parameters [14, 16, 21].

The broad adoption of quantization in CNN accelerators is motivated by the works of [9, 10], amongst others. They showed that quantized networks with appropriate fixed point data formats can obtain accuracies comparable to those of floating point models. Although quantization could be applied to both training and inference, [6] has shown that on-line learning is not necessary for most applications. Thus, inference is where low-power consumption is most important, while training can be left to the cloud.

Brought to the extreme, the quantization idea lead to the development of Binary CNN models (BNNs), i.e. 1-bit quantization [11]. Accelerators for BNNs with remarkable energy footprints are already starting to be designed [3]. However, BNNs are obtaining state-of-the-art performance only on simple tasks (e.g. handwritten digit recognition).

All previous solutions use quantization in an *input-independent way*: the bit-width is set either uniformly for the entire network, or differently for different sections of the model (single neurons or layers), yet independently on the considered input data. None of them considers the possibility of having a *dynamic* tuning of the precision based on input characteristics.

The work in [15], although not resorting to quantization, is one of the first to propose an input-dependent dynamic solution. Borrowing the big/little paradigm in processors, they propose an architecture consisting of *two CNNs of different complexity (and energy consumption)*. In normal conditions, the "little' network is used. However, when the latter is not sufficient to classify a particular input, the "big" network is executed. [17] further improves this concept, by designing the "little" network to be a subset of the "big", hence reducing the overheads associated with two complete models (e.g. memory occupation for weights).

3 MOTIVATION

As mentioned in Section 2, quantization techniques for CNNs are mostly static. The adaptation of existing CNN models to a static quantization method is not trivial, and requires an incremental retraining step [10].

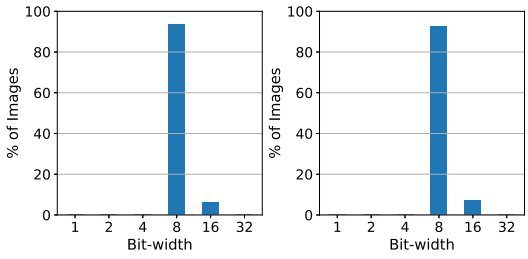
Retraining is costly for many reasons. Besides its sheer computational cost, it often also implies a non-trivial bidwidth "calibration" step. The latter consists of a lengthy iterative analysis, in which, for a given bit-width, the network is retrained, and its accuracy is checked on the validation set. These steps are repeated until the accuracy matches the original floating point model. This is equivalent to considering quantization bit-widths as new hyper-parameters of the network (in addition to the number of layers, number and size of each feature map, etc.).

In contrast, it would be desirable being able to directly execute models already *trained in floating point* on fixed-point accelerators, by simply quantizing the learned weights and the feature maps, without retraining. With static quantization, however, this may

either limit the achievable savings, or cause accuracy drops. This occurs whenever there is a set of inputs that is more "difficult" to classify than the average, at a given bit-width.

An example of this scenario is shown in Figure 2. The two histograms refer to CaffeNet [1], one implementation of the popular AlexNet CNN for image recognition [12], and represent the percentage of images from the ImageNet dataset that can be classified correctly with a given number of bits.

Specifically, the plots are obtained as follows. We first classify *all* images of the training and validation sets using the original floating point network. Then, for each image classified correctly by the original network [1], we quantize *all weights and feature maps* using the method of [10], without re-training. We consider bit-widths in steps of powers of two, starting from 32 bits down to 1 bit. For each image, we record the *minimum bit-width* for which the inference result is correct. The histogram reports the percentage of images that have a given minimum bit-width.



(a) Training set. **(b) Validation set.**

Figure 2: Minimum bit-width for correct classification versus percentage of images in CaffeNet [1].

The figure shows that 8 bits are sufficient for correct inference for most of the images; however, there is approximately a 7% of "difficult" inputs, for which decreasing below 16-bit causes an error in the classification. There are also a negligible number of images that require more than 16-bit or less than 8-bit (bars are not visible). Based on the histograms, if a uniform 8-bit quantization is used for the network, the top-1 accuracy will have a non negligible drop. On the other hand, if 16 bits are used for all images, about 93% of the times we will perform too precise computations and consume unnecessary power.

A CNN capable of dynamically adapting its precision based on inputs is precisely the contribution of this work; our solution will allow to obtain an accuracy that closely matches the floating point reference, while significantly reducing the energy consumption with respect to a "conservative" static quantization (e.g. 16-bit in the above example).

In the following, we assume a uniform quantization, i.e., identical in each layer of the CNN. Previous works have shown that a non-uniform (e.g., per-layer) quantization is generally superior [14]; however, we limit our analysis to the former approach mainly due to its low hardware implementation complexity. In any case, generality is not impaired and our solution can be considered conservative: a non-uniform dynamic quantization could possibly further reduce the bit-width in some layers (e.g. < 8-bit), and thus improve the accuracy/power tradeoff.

[1]We consider the top-1 accuracy metric, i.e. classifications are deemed correct when the network output with the highest probability corresponds to the image label. However, our approach can also be extended to e.g. top-5 accuracy.

4 PROPOSED METHODOLOGY

4.1 Bitwidth vs. Accuracy Characterization

The analysis of the previous section highlights the need for using multiple quantization configurations for different "categories" of inputs. The optimal configurations to be used depend on the network architecture, and can be obtained through a preliminary offline characterization, whose flow chart is shown in Figure 3.

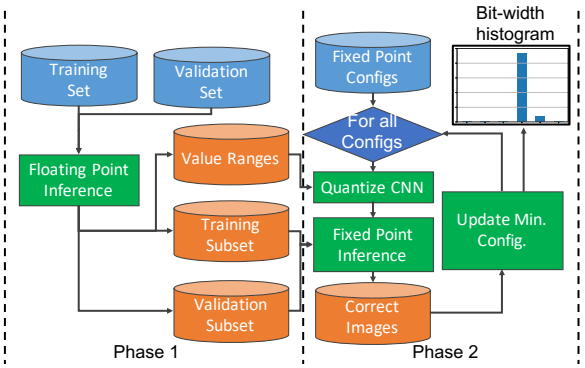


Figure 3: Proposed CNN quantization characterization.

The characterization consists of two phases, shown in the left and right halves of Figure 3 respectively, and its outcome is a histogram similar to the one of Figure 2.

In Phase 1, all images of the training and validation sets are classified using the original double-precision floating point network. The inputs that are correctly classified according to the target metric are stored for later use (*Training/Validation Subsets* in the figure). Moreover, the ranges of values assumed by each learned weight and feature are also recorded (*Value Ranges* in the figure). These are important to determine the number of integer and fractional bits during quantization.

In Phase 2, all available quantization configurations are considered in decreasing order of complexity (i.e. from the largest bit-width to the smallest one). For each configuration, the actual network quantization is performed; in this work, we quantize both weights and feature maps using dynamic fixed-point format and stochastic rounding, as described in [10]. Once the quantized network is available, the previously stored input subsets are classified with it. The output of the network for each image is checked against the reference label, and correct images are recorded. Finally, a table storing the least complex configuration that ensures correct classification for every image is updated. This table is then used to produce the final histogram.

Assuming that a pre-trained model of the original floating point CNN is available, the flow of Figure 3 does not include any further training, and only consists of repeated *inferences*. Even for a complex CNN, the execution time on a workstation equipped with a Titan XP GPU is in the order of 1-2 hours. Moreover, notice that this is a one-time procedure, executed offline before deployment.

Given the results of the above characterization, the applicability of our dynamic bit-width reconfiguration technique can be assessed. In order for our method to be effective, the produced histogram should have a shape similar to Figure 2. Specifically, it should contain *two or more* "relevant" bars (otherwise a static quantization would suffice), and the bar corresponding to the simplest quantization should account for most of the images. The latter corresponds to having a "common case" which can be executed at low precision, and fewer "difficult" inputs that require higher precision.

To obtain the final set of quantization configurations, we post process the output histogram, imposing a minimum height limit of 1%. Bars that are lower than the limit are merged with those relative to higher precisions. This avoids considering quantizations that are very rarely needed, and whose usage would not be beneficial for the method described in Section 4.2. Notice that, after this post processing, for both CNNs considered in our experiments of Section 5, the considered number of quantization configurations is two. However, the generalization of our approach to more than two quantizations is straightforward [17].

4.2 Dynamic Bit-Width Reconfiguration

When the processed histogram is multi-modal (as in the case of CaffeNet) dynamic bit-width reconfiguration can be implemented as shown in Figure 4. With respect to a standard CNN inference using a hardware accelerator, the only addition required is the *Quantization Control* box , which can be implemented easily in software. Therefore, there are *no* hardware overheads in our method.

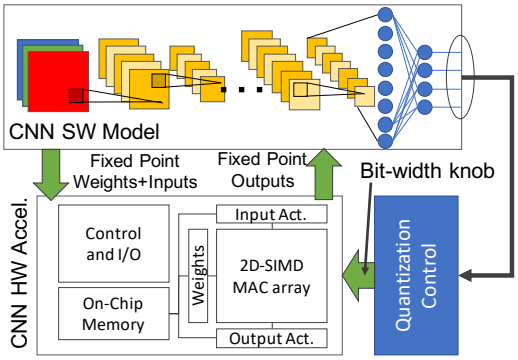


Figure 4: Proposed dynamic bit-width reconfiguration method for CNNs.

The quantized CNN models determined during the characterization phase are stored in main memory. For each new image, the least complex (i.e. smallest bit-width) configuration is initially offloaded to the CNN accelerator memory, and a first classification is performed. Based on the output of the network, the *Quantization Control* block assesses the level of *confidence* of the classification. If the confidence is high enough, the classification output is committed, and the system goes on to process the next image. Otherwise, the network bit-width is increased, and the process is repeated, until the confidence surpasses the desired threshold, or the most complex configuration is executed.

To assess classification confidence, we use the metric proposed in [15] and [17], i.e., the so-called *score margin* (SM):

$$SM = P(y_i|x) - P(y_j|x) \quad (1)$$

where $P(y_k|x)$ is the k-th output of the CNN, representing the probability that the input x belongs to class k, and i and j are the indices of the first and second highest output values. We compare the SM with a threshold T_h, and commit our classification whenever $SM \geq T_h$. Intuitively, the SM measures the difference between the two highest output scores produced by the network; when large, input x has a very high probability of belonging to class i. Conversely, when the difference is small, the input could belong with similar probability to classes i and j.

The larger the decision threshold T_h, the lower the "confidence" of the quantization controller, since a larger SM will be required to consider a classification as correct. Consequently, more classifications per image will be performed on average, with an associated time and energy overhead. Vice versa, the lower T_h, the smaller the overhead, as more classifications will be "accepted", but the accuracy compared to the reference model may decrease if a wrong classification is wrongly deemed correct. Therefore, T_h can be used as a knob to explore the accuracy versus energy tradeoff. We propose to first set the desired accuracy loss (with respect to the floating point model) as an initial constraint, then set T_h to the smallest value that yields the desired accuracy, to minimize overheads.

Notice that, although the score margin method is analogous to the one used in [15] and [17], our approach is significantly different. Indeed, [15] requires *two complete (separately trained) CNN architectures to work*. Both networks work on floating point data, and differ in the number and parameter of layers. In [17], this large overhead is partially reduced by constructing the *little* networks as subsets of the *big* one. However, both papers do not consider the energy benefits deriving from dynamic bit-width reconfiguration.

4.3 Hardware Support

The hardware block diagram in Figure 4 is purposely kept general, to underline that our approach is, in principle, agnostic of the selected CNN accelerator. Nonetheless, in order to benefit from our technique, the hardware must satisfy some basic requirements. Firstly, it must support dynamic reconfiguration of the datapath bit-width from software (*Bit-width knob* in Figure 4). Secondly, reduced bit-width operations must be implemented so that they can provide significant energy reductions.

Among the several accelerators that provide this kind of capability, in this work we focus on the *Envision* chip proposed in [13] and [14]. The peculiarity of this chip is the usage of a technique called Dynamic Voltage, Accuracy and Frequency Scaling (DVAFS). DVAFS consists in combining classic DVFS with sub-word parallel operation to concurrently increase throughput and reduce power. The increased throughput then allows to decrease the operating frequency, for further power benefits [13]. The Envision chip, manufactured in 28nm FDSOI technology, includes a reconfigurable 2D Single Instruction Multiple Data (SIMD) array of Multiply and Accumulate (MAC) operators to implement convolutional layers (see Figure 1). This array can be configured to work at 4,8 and 16-bit precision, and internally leverages DVAFS in order to maximize the power reduction when working at smaller bit-widths. All the rest

Bit-width	Power [mW]
4-bit	7.6
8-bit	56
16-bit	290

Table 1: Power consumption of Envision [14] for different bit-widths, for a constant throughput of 76GOPS.

of the chip is designed to efficiently support multiple precision, e.g. through multi-bank memory, reconfigurable registers, etc.

Although the hardware of Envision is reconfigurable, the original authors use it only for *static* quantization [13, 14]. They use the reconfigurability to change the bit-width on a layer-by-layer basis, but still independently on input data. In [14], they propose to use the chip for *hierarchical classification* (i.e. a sequence of increasingly complex classification tasks). Notice that this is completely different from the proposed method, in which the *same* classification task is performed at different precisions depending on the input.

5 EXPERIMENTAL RESULTS

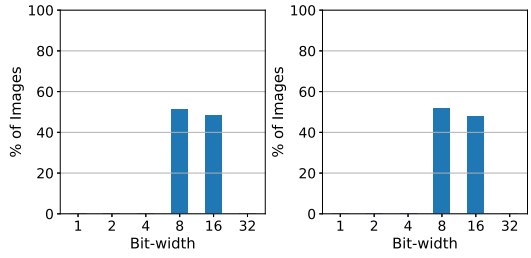
5.1 Setup

To show the results of our methodology, we consider two popular CNN architectures: CaffeNet [1] and CNN-M from [5]. Both networks classify images from the ImageNet dataset, which contains about 1.2M images belonging to 1,000 different classes. We perform inference using the Caffe Deep Learning Framework [1], using pre-trained models of the two networks, downloaded from the Caffe Model Zoo.

The algorithm of Figure 3 is written in Python 2.7 and executed on a desktop workstation (8-thread Intel Core i7 CPU @ 2.67GHz with 8GB RAM, running CentOS 6), equipped with a NVIDIA Titan XP GPU. For energy estimates, we use average power data of the Envision chip (our target CNN accelerator), obtained from [14]. Values are reported in Table 1, and refer to a constant throughput of 76GOPS. For reference, this throughput allows approximately 47 classifications per second on CaffeNet [13].

5.2 Characterization Results

The results of our characterization for the CaffeNet network have been presented in Section 3 and Figure 2. Results for the CNN-M architecture are reported in Figure 5.

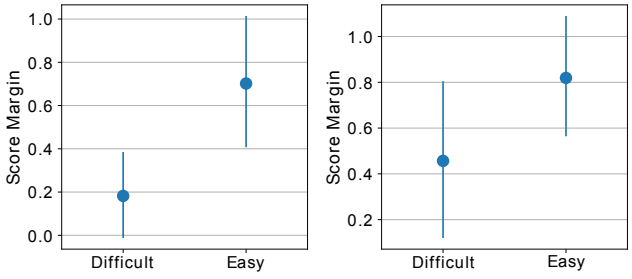


(a) Training set. **(b) Validation set.**

Figure 5: Minimum bit-width for correct classification versus percentage of images in CNN-M [5].

The input dependence of the optimal bit-width for CNN-M is less skewed than in the case of CaffeNet: the two bars have comparable heights. This is a less desirable condition for energy saving in our approach, since a larger number of images will require two inference "iterations" to be classified correctly (first at 8-bit and then at 16-bit). However, it also means that the accuracy degradation for choosing a fixed 8-bit quantization will be much more marked than for CaffeNet. The two distributions of Figures 2 and 5 also suggest that considering more than two quantizations might not be needed for most CNN architectures.

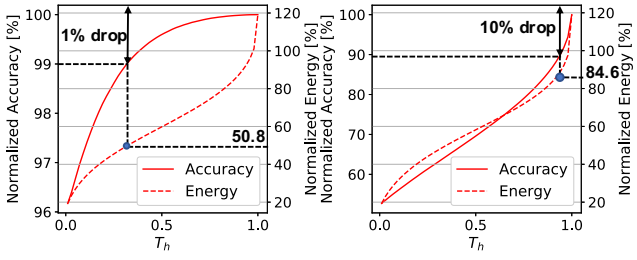
Figure 6 shows the score margin (SM) distributions (means and standard deviation intervals) obtained for all training images with the two networks. Inputs are split into *Easy*, i.e. those that can be correctly classified at 8-bit, and *Difficult*, i.e. those that cannot. The clear difference of means and slight overlap of the intervals confirm that the SM is a good discriminator between reliable and unreliable classification outputs obtained at reduced bit-width. Of course, there are (few) images which can be correctly classified at 8-bit despite a small SM, and vice versa (corresponding to the overlap of the distributions), but on average, the SM method performs well despite its simplicity.



(a) CaffeNet. **(b) CNN-M.**

Figure 6: Score margin for *Easy* and *Difficult* images.

5.3 Accuracy Versus Energy



(a) CaffeNet. **(b) CNN-M.**

Figure 7: Normalized accuracy and energy as a function of the score margin threshold for the two benchmark CNNs.

Figure 7 shows the results of applying our method to the two considered CNNs. Specifically, the plots show the top-1 accuracy and energy consumption of the reconfigurable networks for different values of the Score Margin threshold T_h. Energy points are obtained using the values of Table 1, and are normalized with respect to the

Method	Energy Saving	Top-1 Drop	Multi CNN	Multi Train
[15]	53.7%	0.9%	Yes	Yes
[17]	32.61%	0.29%	No	Yes
Ours	49.2%	0.89%	No	No

Table 2: Comparison of the proposed method with state-of-the-art techniques for CaffeNet.

energy of a static 16-bit quantization. Accuracy is normalized to the results of the baseline floating point networks, which are almost identical to those of a static 16-bit quantization. Both plots refer to the ImageNet validation set.

As expected, results for CaffeNet are characterized by a smaller accuracy variation, due to the skew of the distribution in favor of "easy" inputs. This implies that even for low values of T_h, i.e. when 16-bit inference is performed rarely (hence limiting the power overhead), accuracy quickly approaches the value obtained with static 16-bit. Notice that a pure 8-bit inference without retraining, corresponding to the case of $T_h = 0$ (i.e. all classifications accepted regardless of the *SM*), would result in a $\approx 4\%$ accuracy drop on the validation set, and a power reduction of 78%.

With our approach, it is possible to tune the accuracy versus energy tradeoff. For instance, if only a 1% drop is acceptable ($T_h \approx 0.33$), it is possible to still obtain 49.2% energy reduction compared to the static 16-bit solution.

For CNN-M, due to the greater balance between easy and difficult inputs, the accuracy penalty for choosing a fixed 8-bit quantization is greater (almost half of the baseline). However, it is still possible to reduce the drop to 10% ($T_h \approx 0.89$), and obtain an energy reduction of about 16.4% compared to static 16-bit classification.

5.4 Comparison with the State-of-the-art

Table 2 shows a comparison of our method with the only other two works that propose input-dependent adaptation of CNN inference [15, 17]. Results refer to CaffeNet, which has been evaluated in all three papers. Notice that the energy saving results for [15] and [17] have been obtained on different HW accelerators than what we considered here; we could not compare on the same platform because those accelerators do not support reconfigurable precision operation. Moreover, their power consumption data are not publicly available. Therefore, these numbers should only serve as a rough comparison.

The *Top-1 Drop* refers to the accuracy drop as discussed in the previous section. The table shows that our method obtains results comparable to state-of-the-art approaches in terms of energy saving versus accuracy trade-off. However, it has the important advantage of not requiring multiple network designs, nor multiple training runs. Moreover, notice that dynamic bit-width adaptation is *orthogonal* to those solutions, i.e. it is possible to combine bit-width adaptation with reduction/increase of the number of layers and feature maps, as done in [15, 17], for even greater power savings. The combination of these methods will be the subject of future work.

6 CONCLUSIONS

In this work we addressed the problem of designing energy-efficient deep learning hardware accelerators based on input-dependent, dynamically adaptive re-configuration of their bit-width. The distinctive features of our approach are that (i) it can be applied to popular models without retraining, (ii) the reconfiguration is done on the fly based on the input data and with negligible overhead. Our results show that, when correct classification results can be obtained with at least two different bit-widths, sizable energy reductions are possible. The latter are obviously dependent on the actual shape of the distribution.

Our work described only a first possible embodiment of a general idea, which leaves space to many potential architectural variants that will be the subject of future work, such as a fine-grain quantization for different sections (e.g., layers) of a DNN, or combination with big/little approaches.

REFERENCES

[1] BVLC Caffe: https://github.com/BVLC/caffe.
[2] M. Alioto. Energy-quality scalable adaptive VLSI circuits and systems beyond approximate computing. *DATE 2017*, pp. 127–132.
[3] R. Andri et al. YodaNN: An ultra-low power convolutional neural network accelerator based on binary weights. *IEEE ISVLSI 2016*, pp. 236–241.
[4] K. Bong et al. Low-Power Convolutional Neural Network Processor for a Face-Recognition System. *IEEE Micro 2017*, 37(6):30–38.
[5] K. Chatfield et al. Return of the Devil in the Details: Delving Deep into Convolutional Nets. *arXiv:1405.3531*, 2014.
[6] Y. Chen et al. DaDianNao: A Machine-Learning Supercomputer. *IEEE Microi 2015*, pp. 609–622.
[7] Y. H. Chen et al. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. *IEEE JSSC*, 52(1):127–138, 2017.
[8] I. Goodfellow et al. *Deep Learning*. MIT Press, 2016.
[9] S. Gupta et al. Deep Learning with Limited Numerical Precision. *ICML 2015*, pp. 1737-1746.
[10] P. Gysel. Hardware-Oriented Approximation of Convolutional Neural Networks. *arXiv:1604.03168*, 2016.
[11] I. Hubara et al. Binarized Neural Networks. *Advances in Neural Information Processing Systems 29*, pp. 4107–4115, 2016.
[12] A. Krizhevsky et al. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, 2012.
[13] B. Moons et al. DVAFS: Trading computational accuracy for energy through dynamic-voltage-accuracy-frequency-scaling. *DATE 2017*, pp. 488–493.
[14] B. Moons et al. Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network processor in 28nm FDSOI. *IEEE ISSCC 2017*, pp. 246–247.
[15] E. Park et al. Big/little deep neural network for ultra low power inference. *CODES+ISSS 2015*, pp. 124–132.
[16] B. Reagen et al. Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators. *ACM/IEEE ISCA 2016*, pp. 267–278.
[17] H. Tann et al. Runtime configurable deep neural networks for energy-accuracy trade-off. *IEEE/ACM/IFIP CODES 2016*, pp. 1–10.
[18] S. Venkataramani et al. Scalable-effort classifiers for energy-efficient machine learning. *DAC 2015*, pp. 1–6.
[19] S. Venkataramani et al. AxNN: Energy-efficient Neuromorphic Systems Using Approximate Computing. *ISLPED 2014*, pp. 27–32.
[20] C. Zhang et al. Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks. In *ACM/SIGDA FPGA 2015*, pp. 161–170.
[21] Q. Zhang et al. ApproxANN: An approximate computing framework for artificial neural network. *DATE 2015*, pp. 701–706, 2015.