Quantization of constrained processor data paths applied to convolutional neural networks

DOI:
10.1109/DSD.2018.00069

Document status and date:
Published: 12/10/2018

Document Version:
Accepted manuscript including changes made at the peer-review stage

Quantization of constrained processor data paths applied to Convolutional Neural Networks

Barry de Bruin
Eindhoven University of Technology
Eindhoven, The Netherlands
Email: e.d.bruin@tue.nl

Zoran Zivkovic
Intel
Eindhoven, The Netherlands
Email: zoran.zivkovic@intel.com

Henk Corporaal
Eindhoven University of Technology
Eindhoven, The Netherlands
Email: h.corporaal@tue.nl

***Abstract*—Artificial Neural Networks (NNs) can effectively be used to solve many classification and regression problems, and deliver state-of-the-art performance in the application domains of natural language processing (NLP) and computer vision (CV). However, the tremendous amount of data movement and excessive convolutional workload of these networks hampers large-scale mobile and embedded productization. Therefore these models are generally mapped to energy-efficient accelerators without floating-point support. Weight and data quantization is an effective way to deploy high-precision models to efficient integer-based platforms. In this paper a quantization method for platforms without wide accumulation registers is being proposed. Two constraints to maximize the bit width of weights and input data for a given accumulator size are introduced. These constraints exploit knowledge about the weight and data distribution of individual layers. Using these constraints, we propose a layer-wise quantization heuristic to find a good fixed-point network approximation. To reduce the number of configurations to consider, only solutions that fully utilize the available accumulator bits are being tested. We demonstrate that 16-bit accumulators are able to obtain a Top-1 classification accuracy within 1% of the floating-point baselines on the CIFAR-10 and ILSVRC2012 image classification benchmarks.**

***Index Terms*—quantization, fixed-point, efficient inference, narrow accumulators, convolutional neural networks**

I. Introduction

Neural Networks (NNs) are a class of machine-learning algorithms that deliver state-of-the-art performance on many natural language processing (NLP) (e.g. speech recognition and natural language understanding) and Computer Vision (CV) tasks, such as object localization, classification, and recognition of objects. Unfortunately, both the training and inference phase of the NNs are computationally demanding. As a result, NNs are usually developed and trained on high-performance clusters, whereafter the resulting model (learned weights) is mapped to an optimized hardware platform for efficient model deployment on mobile and embedded devices. These hardware platforms generally use reduced-precision integer arithmetic, which simplifies the data path and reduces memory storage, bandwidth, and energy requirements.

The process of converting a pre-trained floating-point NN model to reduced-precision is called quantization. Typical quantization procedures for NNs check a number of possible reduced-precision solutions for both weights and intermediate data within a given network, and choose the cheapest minimal bit width solution within a tolerable model accuracy penalty. Due to the enormous set of potential quantization solutions, it is not feasible to find an optimal solution. Typically just a small number of solutions are evaluated. We identify that the bit width of partial result accumulators is generally a bottleneck on platforms that are not specifically optimized for large kernel computations. Kernel computations are the core of many NN architectures, such as traditional fully-connected NNs, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

This accumulator bottleneck further complicates the quantization procedure. To address this issue, we formulate the NN quantization problem as a function of accumulator size. For a fixed accumulator bit width, we aim to maximize the model accuracy by maximizing the bit width of input data and weights. To increase the maximum data and weight bit width, we introduce two constraints that provide a more optimistic maximum accumulator range estimate, while still providing analytical guarantees on avoiding potential accumulator overflow. The main contributions of this paper are:

- A new quantization method that considers a limited accumulator size. This is very useful for platforms with narrow accumulators.
- A heuristic for fast quantization of complete CNNs for image classification.
- An evaluation of our layer-wise quantization heuristic on three popular CNN benchmarks.

The rest of the paper is organized as follows: Section 2 summarizes related work in CNN quantization. Section 3 covers preliminaries and introduces notation and background for fixed-point inference. Section 4 introduces the quantization method for accumulator-constrained accelerators. Section 5 explains the heuristic for layer-wise CNN quantization. Evaluation and experimental results are provided in Section 6. Concluding remarks follow in Section 7.

II. Related Work

Recent works have investigated the feasibility of converting floating-point NNs to fixed-point for efficient implementation onto mobile and embedded devices. It has been well-established that NNs are generally very resilient to quantization noise and do not require large bit widths for good performance. Courbariaux, David and Bengio [1] train a CIFAR-10 network with a negligible accuracy penalty, using only 10 bits for weights and activations (and 12 bits for weight updates). A primary innovation is the application of a dynamically scaled fixed-point format. Their training procedure monitors if overflow happens, and adjusts the scaling factors of weights and data in NN layers accordingly. FlexPoint [2] extends on this work by providing hardware support for dynamic overflow-based scaling factor management. Their custom 16-bit floating-point format with shared exponents matches the classification accuracy of 32-bit floating-point baselines on several large NN benchmarks. Rastegari et al. [3] obtains respectable classification accuracy on the difficult ILSVRC2012 benchmark with only 1-bit weights.

The downside of the previous quantization methods is that the training procedure of large NNs can be very time-consuming. For many use cases it suffices to retrain a pre-trained model on your own (similar) dataset [4]. Many related works [5]–[7] do therefore focus on quantization of a pre-trained model within a tolerable accuracy penalty. To regain some of the lost accuracy during quantization, the quantized model is generally retrained [5]–[9].

Vanhoucke et al. [10] linearly normalizes weights and (sigmoid) activations of every layer in a speed-recognition NN to 8-bit by analysing the range of weights and activations. A similar approach is implemented in several deep learning frameworks such as Tensorflow [11] and Caffe-Ristretto [12]. Lin, Talathi, and Annapureddy [6] propose an analytical model to quickly convert pre-trained models to fixed-point. The advantage of this model is that it does not require exhaustive layer-wise optimization, but rather determines the bit width of every layer based on the approximate parameter and data range distribution.

An exhaustive layer-wise optimization is a straight-forward approach for NN quantization. This procedure generally consists of testing many possible quantization solutions for every layer in the network [7], [9]. To reduce the number of solutions to consider, several heuristics were developed. Gysel et al. [12] propose an iterative quantization procedure where weights are quantized first, and activations are quantized second. A similar two-step approach is described by other related works [13]. Shan et al. [14] considers some target platform characteristics, and shows that the accumulator bit width can be reduced to 16-bit without significant a significant penalty in classification accuracy for the LeNet5 benchmark.

We consider a target platform that is not specifically optimized for large NN kernels and does not have a wide accumulator data path. Knowing that the accumulator is the bottleneck, we define several constraints that greatly reduce the number of solutions to consider. The quantization problem is reformulated to maximize the model accuracy for a fixed accumulator bit width.

III. Background

A. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of NNs that work very well for a variety of Computer Vision tasks. They can be used as an end-to-end approach for object classification, localization, tracking, or even higher-order tasks such as object recognition. CNNs are composed of layers that are generally connected in a feed-forward fashion. The input of the first layer is an (preprocessed) image. The model output depends on the task, and can be a class likelihood vector for image classification, a bounding box for object localization, or a combination of both.

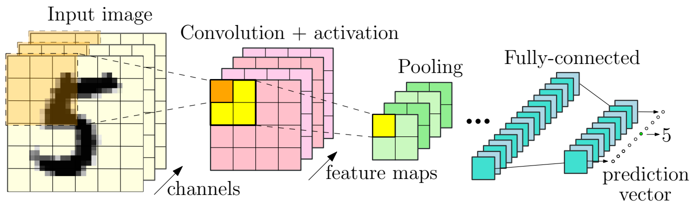

Fig. 1. Example CNN for handwritten digit classification.

A typical CNN consists of several stacks of convolution + activation + pooling (+ normalization) layers for feature extraction, followed up by several fully-connected (+ activation) layers for classification. The main computation of a CNN are the convolution and fully-connected layers. Both layer types can be simplified to a series of multiply-accumulate operations or dot product:

$$y = \sum_{i=1}^{K} w_i d_i \tag{1}$$

where w_i are the learned weights (and bias), d_i are data values, y is the output result of a single neuron before activation, and K is the kernel size.

B. Fixed-point preliminaries

NN models are generally trained using the single-precision floating-point format. To approximate these models on a target platform that only supports 2's complement integer arithmetic, we use a fixed-point number representation. This format can be represented as a combination of Bit Width (BW), Integer (Word) Length (IL), Fractional (Word) Length (FL), and a sign bit:

$$BW_{\boldsymbol{x}} = IL_{\boldsymbol{x}} + FL_{\boldsymbol{x}} + 1 \tag{2}$$

where $\boldsymbol{x}$ corresponds to a set of floating-point values that share the same fixed-point representation. Any real value $x \in \boldsymbol{x}$ can be quantized to its integer representation by scaling and rounding:

$$Q(x) = round(x \cdot 2^{FL_{\boldsymbol{x}}}) \tag{3}$$

The original real value can be estimated back by scaling the integer values back:

$$\hat{x} = Q(x) \cdot 2^{-FL_{\boldsymbol{x}}} \tag{4}$$

The maximum quantization error is bounded by the choice of rounding function. For example, if we consider a round-to-nearest policy, the quantization error is bounded by half of the least significant digit e.g. $2^{-(FL_{\boldsymbol{x}}+1)}$. To minimize this quantization error the fractional length or scaling factor should be maximized. However, increasing the fractional length of a fixed-point format will reduce the representable range:

$$-2^{IL_{\boldsymbol{x}}} \leq \hat{x} \leq 2^{IL_{\boldsymbol{x}}} - 2^{-FL_{\boldsymbol{x}}} \tag{5}$$

If data values fall outside the range of our 2's complement integer representation, we either need to clip these values or overflow will happen. Therefore we must ensure that the integer length is chosen large enough to prevent overflow:

$$IL_{\boldsymbol{x}} = \lfloor \log_2 (R_{\boldsymbol{x}}) \rfloor + 1 \tag{6}$$

where range $R_{\boldsymbol{x}}$ corresponds to the absolute maximum value within set $\boldsymbol{x}$ i.e.

$$R_{\boldsymbol{x}} = \max_{x \in \boldsymbol{x}} |x| \tag{7}$$

For example, consider a fixed-point group $\boldsymbol{x}$ with $BW_{\boldsymbol{x}} = 16$ and a maximum range of $R_{\boldsymbol{x}} = 0.1256$. From Equation 6 follows that we require at least $IL_{\boldsymbol{x}} = -2$ to prevent overflow. Stated differently, maximum precision can be obtained by scaling all values in group $\boldsymbol{x}$ by $2^{FL_{\boldsymbol{x}}}$ for $FL_{\boldsymbol{x}} = 17$, using Equation 2.

The range estimate of Equation 7 has been used by others [5], and worked very well for our experiments. Other [6], [15], more optimistic, range estimates were also considered, but without much success; results of different range estimates were inconsistent between different benchmark. These findings indicate that having sufficient range is more important than precision, which is in line with the work of Lai et al. [16].

C. Range Analysis

A common approach to determine $R_{\boldsymbol{x}}$ is to analyse the range of weights and input data [7], [11], [12], [14] for every convolutional or fully-connected layer within a NN. The weight range is extracted from the high-precision pre-trained model. The data range can be estimated by forwarding a large batch of images through the network. Fig. 2 visualizes a typical density distribution of a layer from the popular AlexNet [17] network. It can be observed that weights are significantly smaller than input data, but that both groups are clustered together. Additionally, the range between layers also differs significantly, as is depicted in Fig. 3.

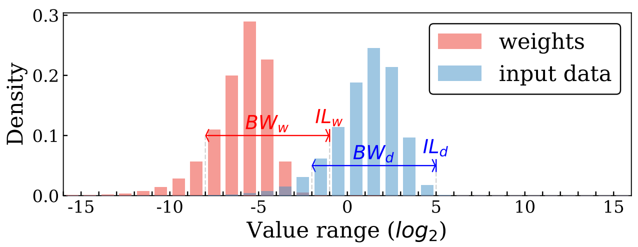

Fig. 2. Value range of weights (red) and input data (blue) of AlexNet's third convolution layer. The two intervals suggest a potential quantization solution. Values outside the intervals cannot be represented. Zero-valued data is ignored in this distribution; Zero can always be represented (see Equation 5).

Using a single fixed-point format for the whole network is not optimal due to the large range difference between different groups and layers. To encode this network in fixed-point, we require an integer length that is large enough to prevent overflow, while the precision or fractional length should be large enough to distinguish variations in weights.

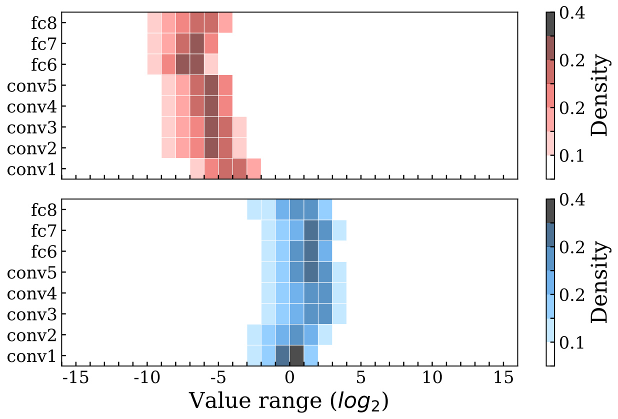

Fig. 3. Value range of weights (top) and input data (bottom) of AlexNet's convolutional and fully-connected layers (conv3 corresponds to Fig. 2).

In NNs this dynamic range problem can be addressed by splitting weights and input data within a layer into separate *fixed-point groups* [18]. A *fixed-point group* is a set of values that share the same bit width and fractional length. In other words, the global scaling factor is replaced by multiple local scaling factors. An example of this is shown in Fig. 2, where both weights and data have a separated integer length and bit width to optimally capture the data range.

Using different fixed-point groups for both weights and data in layer adds minimal computational overhead. Weights can be quantized off-line and integer multiplication of two fixed-point groups with different scaling factors requires no precision alignment. Additionally, we do not reduce precision of intermediate results during kernel computation. Results of a previous layer that are used as input for the next layer might require precision alignment. This procedure consists of a single arithmetic shift with clipping and rounding, which is negligible, when we consider that every input sample can generally be reused for 100–1000s multiply-accumulations.

IV. Quantization with narrow accumulators

In the previous section we introduced the relevant concepts for fixed-point NN inference and motivated the use of multiple fixed-point groups. In this section we define our quantization methodology for convolutional and fully-connected layers for a data path with limited data bus and accumulator size.

Consider a target platform with a data bus bit width BW_{data} and accumulator bit width BW_{acc}. These platform-dependent parameters restrict the bit width of weights ($BW_{\boldsymbol{w}}$) and input data ($BW_{\boldsymbol{d}}$):

$$1 \leq BW_{\boldsymbol{w}} \leq BW_{data} \qquad 1 \leq BW_{\boldsymbol{d}} \leq BW_{data} \tag{8}$$

For efficiency reasons we do not consider data types that are wider than the data bus. Combinations of $BW_{\boldsymbol{w}}$ and $BW_{\boldsymbol{d}}$ are also bounded by the accumulator size:

$$(BW_{\boldsymbol{w}} + BW_{\boldsymbol{d}} - 1) + \Delta \leq BW_{acc} \tag{9}$$

where Δ is a layer-dependent parameter that corresponds to the number of additional integer bits the accumulator requires to store the kernel computation.

A generic quantization approach of a convolutional or fully-connected layer can be summarized by the following steps [5]–[7]:

1) Analyse the parameter and input data range to determine integer lengths $IL_{\boldsymbol{w}}$ and $IL_{\boldsymbol{d}}$. The available bit widths $BW_{\boldsymbol{w}}$ and $BW_{\boldsymbol{d}}$ determine the precision $FL_{\boldsymbol{w}}$ and $FL_{\boldsymbol{d}}$ (see Equation 2).
2) Check if the chosen solution does not overflow the accumulator (see Equation 9).
3) Adapt the layer to the new configuration, and check if the network performance is still sufficient.

To test which combination of $BW_{\boldsymbol{w}}$ and $BW_{\boldsymbol{d}}$ provides the best model accuracy, we must decrease the bit width of groups $\boldsymbol{w}$ and $\boldsymbol{d}$ until the accumulator will not overflow. Testing every combination of $BW_{\boldsymbol{w}}$ and $BW_{\boldsymbol{d}}$ would be very time-consuming. Instead, we define several constraints that provide an upper bound on the maximum accumulator range.

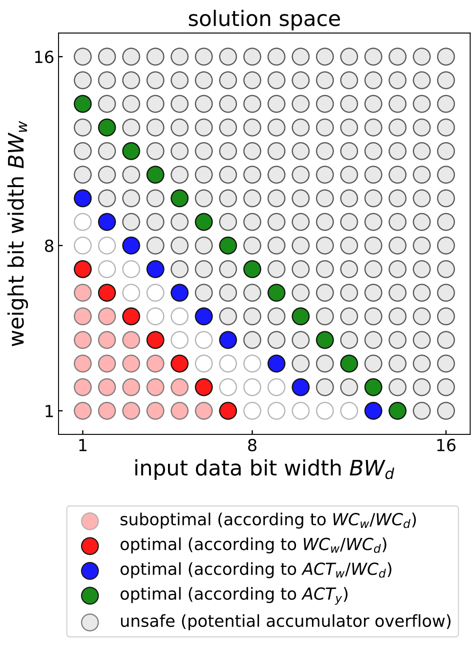

Fig. 4. Illustration of the solution space of a convolutional or fully-connected layer. Red dots are solutions that utilize all accumulator bits for the worst-case constraint. Light red dots are suboptimal as these do not utilize all accumulator bits. Blue dots represent optimal solutions for the ACT_w/WC_d constraint. For small bit widths most weights are quantized to zero, which explains the distortion at $BW_w < 4$. Green represents the optimal solutions according to constraint ACT_y. However, these solutions may result in accumulator overflow.

1) WC_w/WC_d: The most pessimistic constraint considers worst-case input and weight values for every multiply-accumulation. The accumulation register must be large enough to store K kernel accumulations of $w_i d_i$ i.e.

$$(BW_{\boldsymbol{w}} + BW_{\boldsymbol{d}} - 1) + \lceil \log_2 K \rceil \leq BW_{acc} \tag{10}$$

Using Equation 10 we can calculate which bit width combinations of weights and input data optimally utilize the accumulator:

$$BW_{\boldsymbol{w}} + BW_{\boldsymbol{d}} = BW_{acc} + 1 - \lceil \log_2 K \rceil \tag{11}$$

By only considering solutions that fully utilize the accumulator bit width, the number of quantization solutions that need to be tested per layer is reduced from a 2-dimensional search to an 1D search, and is usually very small. For example, if we consider a target platform with a 16-bit accumulator and a kernel $K = 5 \times 5 \times 16$, only a few configurations for $BW_{\boldsymbol{w}}$ and $BW_{\boldsymbol{d}}$ need to be considered, as is illustrated in Fig. 4.

2) ACT_w/WC_d: By using knowledge about our kernel weights, we can derive a more optimistic bound on the maximum accumulator range R_{acc} [19], using the fact that the sum of products is at most equal to the sum of absolute products:

$$R_{acc} = \sum_{i=1}^{K} w_i d_i \leq \sum_{i=1}^{K} |w_i||d_i| \tag{12}$$

To guarantee that this bound will not overflow the accumulator, we need to compute the maximum kernel range including quantization effects (i.e. rounding errors). Additionally, we need to assume worst-case input data for every multiply-accumulate. This results in a simple definition of the maximum accumulator range:

$$R_{acc} = R_{kernel} \cdot 2^{IL_{\boldsymbol{d}}} \tag{13}$$

where kernel range R_{kernel} corresponds to the maximum sum of absolute quantized weights over all kernels within a NN layer:

$$R_{kernel} = \max_{kernels \,\in\, layer} \left(\sum_{i=1}^{K} |\hat{w}_i| \right) \tag{14}$$

R_{kernel} depends on the scaling factor $FL_{\boldsymbol{w}}$ and rounding policy (see Equation 4). We assume a round-away-from-zero tie-breaking policy, and pre-compute a table with the kernel range R_{kernel} as a function of $FL_{\boldsymbol{w}}$ for a given layer.

We will now use R_{acc} to compute IL_{acc} (Equation 6). Using IL_{acc} we can now derive our accumulator constraint BW_{acc} where we assume that the precision of products is not being reduced during kernel computation:

$$\begin{aligned} BW_{acc} &= FL_{\boldsymbol{w}} + FL_{\boldsymbol{d}} + IL_{acc} + 1 \\ &= FL_{\boldsymbol{w}} + BW_{\boldsymbol{d}} + \lfloor \log_2 (R_{kernel}) \rfloor + 1 \end{aligned} \tag{15}$$

Now replace $FL_{\boldsymbol{w}}$ by $BW_{\boldsymbol{w}} - IL_{\boldsymbol{w}} - 1$ (Equation 2) and reorder the terms. This results in the following constraint:

$$BW_{\boldsymbol{w}} + BW_{\boldsymbol{d}} = BW_{acc} - \lfloor \log_2 (R_{kernel}) \rfloor + IL_{\boldsymbol{w}} \tag{16}$$

This constraint will typically give more available bits than the WC_w/WC_d constraint, as weights are generally very small. Since we did include worst-case representable input data and quantized kernel values, the accumulator will never overflow.

3) $ACT_{\boldsymbol{y}}$: An even more optimistic, but potentially unsafe, accumulator constraint can be derived by exploitation of the wrap-around property of 2's complement integer arithmetic. Note that this constraint does not prevent the intermediate accumulator result from overflowing. This is not an issue as long as the final output result fits within valid accumulator range. The maximum output range $IL_{\boldsymbol{y}}$ was already estimated during the range analysis step, which limits the precision of $FL_{\boldsymbol{w}}$ and $FL_{\boldsymbol{d}}$ as follows:

$$BW_{acc} = FL_{\boldsymbol{w}} + FL_{\boldsymbol{d}} + IL_{\boldsymbol{y}} + 1 \tag{17}$$

With some rewriting this results in the following constraint for valid combinations of $BW_{\boldsymbol{w}}$ and $BW_{\boldsymbol{d}}$:

$$BW_{\boldsymbol{w}} + BW_{\boldsymbol{d}} = BW_{acc} + 1 - \max(0, IL_{\boldsymbol{y}} - (IL_{\boldsymbol{w}} + IL_{\boldsymbol{d}})) \tag{18}$$

It should be emphasized that the choice of $IL_{\boldsymbol{y}}$ is data-dependent and should therefore be chosen rather pessimistic to prevent accumulator overflow. The max-operator is necessary to ensure that the result of a single multiplication will fit into the accumulator.

In the next section we will extend this layer-wise quantization method with a heuristic to find a quantization solution for a complete network.

V. Heuristic layer-wise optimization

In this section we will describe the quantization heuristic for complete CNNs. The goal of this procedure is to maximize the Top-1 classification accuracy for a fixed accumulator size and maximum data bit width.

Every convolution and fully-connected layer has a set of feasible quantization solutions. In Section IV we already proposed several accumulator constraints to reduce the solutions space for a given layer. However, testing every possible combination for every layer would still be very time-consuming. Instead we use a straightforward heuristic which iteratively quantizes the network. We start at the first eligible layer at the input of a pre-trained high-precision network, test all solutions that were proposed by the accumulator constraint, set the layer to the best solution for $BW_{\boldsymbol{w}}$ and $BW_{\boldsymbol{d}}$, and repeat the process for the next eligible layer. This process continues until all convolution and fully-connected layers are quantized. Similar to [18], [20] we simulate a fixed-point data path with limited precision. This approach makes it very easy to set a subset of layers to reduced-precision, while the remainder remains high-precision.

The quality of a solution in a given layer is evaluated in terms of Top-1 classification accuracy on a small validation dataset. If two quantization configurations within the same layer result in the same accuracy, we pick the solution that minimizes the Sum of Absolute Residuals (SAR). In other words, we compare the outputs of the floating-point layer to the outputs of the quantized layer, and pick the configuration that minimizes

$$\sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{19}$$

where N equals the number of output samples within the current layer, and y and $\hat{y}$ denote the floating-point and quantized output results, respectively.

The complete procedure for a 4-layer network is visualized in Fig. 5. The quality of the final quantization result is measured in terms of Top-1 classification accuracy on a large separate test dataset. Note that different layers have a different number of available bits for $BW_{\boldsymbol{w}} + BW_{\boldsymbol{d}}$, depending on the data and parameter range and kernel size. In this example layer conv1 has 16 bits to distribute between $BW_{\boldsymbol{w}}$ and $BW_{\boldsymbol{d}}$, while layer conv2 has only 14 bits.

VI. Results and Evaluation

In this section we will evaluate the proposed quantization method from Section IV on 3 popular CNN benchmarks for image classification. In particular, we evaluate the effectiveness of the different accumulator constraints in comparison

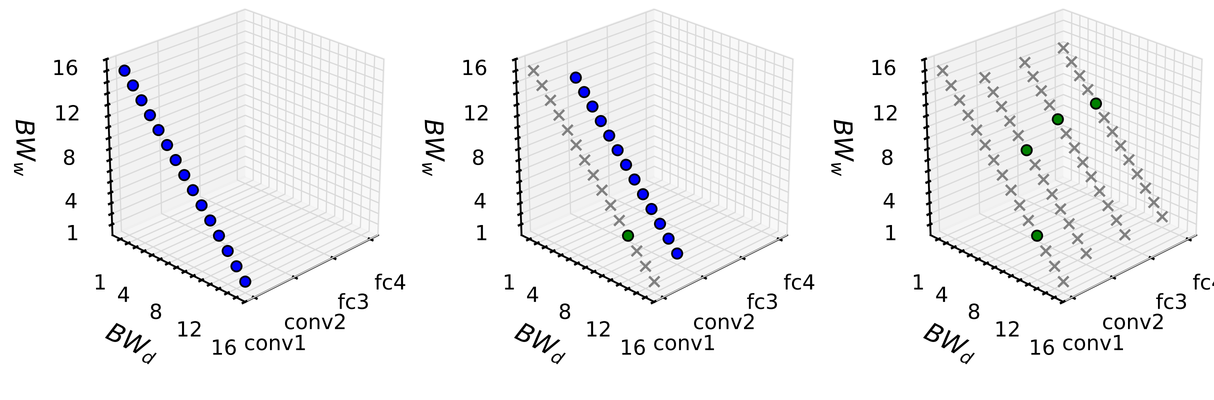

Fig. 5. Quantization procedure for a 4-layer network. Left: Analyse the chosen accumulator constraint for the first layer and test every eligible solution (blue dots). Middle: Set the previous layer to best-performing solution (green dot) and continue quantization of the next layer. Right: Repeat the procedure until all layers are quantized.

to floating-point baselines. We conclude by investigating the optimality of the accumulator constraints in comparison to an optimistic lower bound.

All experiments were performed on an Ubuntu 17.10 machine with 16GB RAM, an Intel-i5 7300HQ, and a GTX1050 with 4GB VRAM. The quantization procedure, as described in Section V was implemented in PyTorch 0.3.1 and accelerated by the GPU using CUDA 8.0 with cuDNN v6. Quantized layers were simulated by reducing the precision and range of data and weights before and after the kernel computation. Overflow behaviour in the accumulator was simulated as well. For simplicity, we do only quantize convolutional and fully-connected layers. Activation, subsampling, and normalization layers are not quantized. However, these layers do generally not have a significant impact on the total CNN workload.

A. Baseline Networks

1) LeNet5: We evaluate our procedure first on a small 5-layer LeNet5(-like) network. This 5-layer CNN is trained on the 10-class MNIST dataset for handwritten digit classification. The network structure is depicted in Table I. All layers (except the last) are followed up by a ReLU activation function. Convolutional layers are also sub-sampled by a 2×2 Max-pooling filter with stride 2. 200 images were sampled from the official MNIST test set for optimization. The remaining images were used to test the final solution. Floating-point results were obtained using the same test set.

TABLE I
LeNet5 network structure

Layer	Kernel size (+ bias)	Output size
input	-	$28 \times 28 \times 1$
conv1	$5 \times 5 + 1$	$24 \times 24 \times 16$
conv2	$5 \times 5 \times 16 + 1$	$8 \times 8 \times 16$
fc3	$32 \times 4 \times 4 + 1$	$4 \times 4 \times 16$
fc4	$512 + 1$	10

2) All-CNN-C: This 9-layer network [21] is trained on the popular CIFAR-10 dataset. We use the pre-trained model from the Nervana model zoo[1]. Similar to LeNet5, 200 images were randomly sampled from the official test dataset for optimization. The remaining 9800 images were used to test the final solution. This network contains layers with larger kernels (e.g. >1000 multiply-accumulations) and is therefore potentially harder to quantize for a platform with narrow accumulators.

3) AlexNet: To investigate the limits of our quantization approach, we also quantized the well-known 8-layer AlexNet CNN [17]. This network is trained on the difficult 1000-class ImageNet ILSVRC2012 dataset. We use the pre-trained model from the PyTorch model zoo[2]. 1000 images were randomly sampled from the official 50K images validation set for optimization. The rest was used to validate the quantized solution. Similar to All-CNN-C, this network contains several layers with large kernels.

B. Maximizing accuracy for different accumulator sizes

Fig. 6 visualizes the quantization results for different accumulator sizes. For this experiment the maximum data bit width BW_{data} is set equal to BW_{acc}. For all three benchmarks a 32-bit accumulator yields no loss in Top-1 classification accuracy, compared to the floating-point baseline. As expected, the WC_w/WC_d constraint performs worse than the other constraints, since its pessimistic accumulator range estimate limits the precision within the kernel computation. Although the ACT_y constraint is potentially unsafe, it matches or outperforms the other two constraints on all benchmarks for different accumulator bit widths. For the small LeNet5 benchmark the accumulator bit width can be reduced to 12-bit, before the relative error increases by more than 20%. For the other benchmarks the accumulator bit width can be reduced to 16-bit within a 5% relative error penalty.

C. Maximizing accuracy for various data bit widths

In the previous experiment the maximum data bit width BW_{data} was fixed to BW_{acc}. However, to satisfy the accumu-

[1]https://gist.github.com/nervanazoo
[2]https://github.com/pytorch/vision

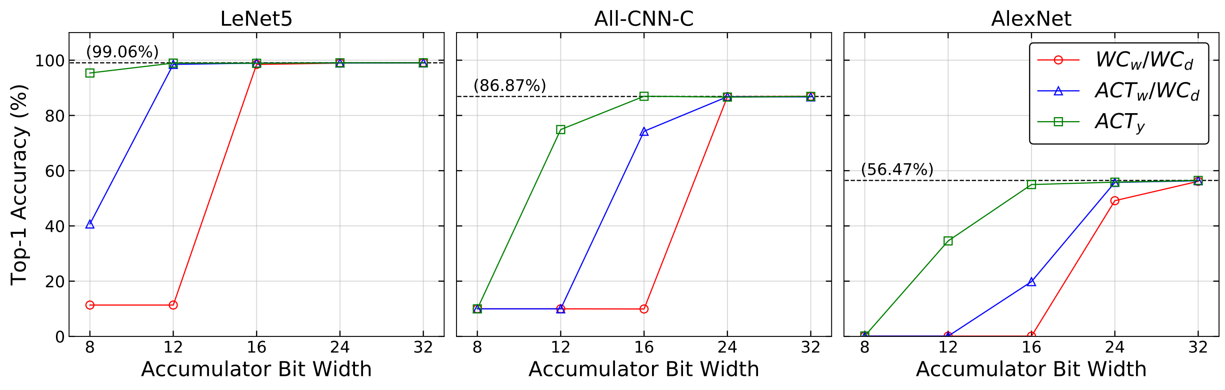

Fig. 6. Quantization results for various accumulator bit width for the 3 constraints. The dashed line indicates the floating-point reference accuracy. It is clearly visible that the most relaxed constraint (ACT_y) gives the highest accuracy.

TABLE II

QUANTIZATION RESULTS (TOP-1 ACCURACY) FOR RESTRICTED DATA AND ACCUMULATOR BIT WIDTH (USING CONSTRAINT ACT_y).

BW_{acc}	32					24				16		
Max. BW_{data}	32	24	16	12	8	24	16	12	8	16	12	8
LeNet5 (99.06%)	99.07%	99.06%	99.06%	99.06%	99.03%	99.06%	99.06%	99.06%	99.03%	98.89%	98.86%	99.06%
All-CNN-C (86.87%)	86.89%	86.87%	86.86%	86.89%	86.72%	86.65%	86.90%	86.94%	86.72%	86.94%	85.96%	86.97%
AlexNet (79.03%)	79.08%	79.05%	79.03%	79.06%	77.86%	78.71%	79.10%	79.11%	77.86%	78.29%	78.23%	77.62%

lator constraints, $BW_{\boldsymbol{w}}$ and $BW_{\boldsymbol{d}}$ are generally set to values way smaller than BW_{data}. As a consequence, many bits are left unused on a typical general-purpose platform that only supports several bit widths. Therefore we have investigated the obtainable classification accuracy for different choices for BW_{data}. In this experiment we only used the most optimistic ACT_y constraint. The results are listed in Table II. For a sufficiently large accumulator (i.e. 32-bit), 16-bit data yields no loss in accuracy for any of the benchmarks. This is in line with other works [5], [7]. From the results follows that the accumulator of all benchmarks can be reduced to 24-bit and 12-bit data without a performance penalty. For 8-bit data types, the classification accuracy of AlexNet is expected to drop by 1–2% [12]. This penalty only increases marginally when the accumulator is reduced from 32-bit to 16-bit.

Overall these experiments demonstrate that large CNNs can run effectively on platforms without support for wide accumulators. Additionally, it has been shown that the proposed quantization heuristic for layer-wise optimization obtains reasonable solutions for narrow accumulators for a variety of maximum data bit widths.

D. Analysis of accumulator constraints

For all previous experiments the accumulator range was chosen such that the chances on potential overflow are minimized (or completely avoided). However, some overflow might be tolerable. This experiment aims to provide insights on an optimistic lower bound on the minimum accumulator range.

We start with a high-precision network and reduce the representable accumulator range (i.e. IL_{acc}) of a single layer. We then forward a small dataset (200 images) through the modified network and observe the Top-1 classification accuracy. Two types of overflow-behaviour were considered: wrap-around and clipping. The results for LeNet5 are depicted in Fig. 7. It follows from the figure that the accuracy drops very steeply when the accumulator has insufficient range. Clipping seems to delay this accuracy breakdown point by at least 1–2 bits of additional accumulator range.

To compare the minimal required accumulator range to the solutions that were found by the accumulator constraints from Section IV, two metrics have been defined: LB_{wrap} and LB_{clip}. LB_{wrap} denotes the minimum integer length of the accumulator for which the normalized classification accuracy is still above 99% for the wrap-around case. Similarly, LB_{clip} denotes the minimum integer length for the case where clipping is used. For the LeNet5 benchmark the required integer lengths are listed in Table III. The optimistic ACT_y constraint

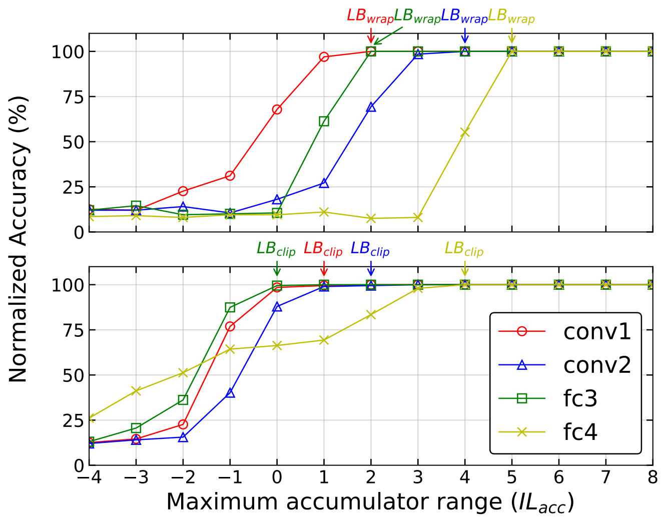

Fig. 7. LeNet5 normalized Top-1 accuracy on small (200-image) dataset when the accumulator range (i.e. IL_{acc}) of a single layer is restricted. Overflow behaviour is either wrap-around (top) or clipping (bottom). For every layer the minimum range before accuracy collapses is indicated.

is generally very close to the point where the network accuracy starts to collapse (LB_{wrap}). If the accumulator would clip instead of wrap-around, we could potentially reduce IL_{acc} by another 1 or 2 bits. The two safe constraints are too pessimistic, which results in suboptimal network accuracy for accumulators below 32-bit (see Fig. 6).

TABLE III

REQUIRED ACCUMULATOR RANGE (IL_{acc}) FOR LENET5 BENCHMARK WITH DIFFERENT CONSTRAINTS.

Layer	WC_w/WC_d	ACT_w/WC_d[a]	ACT_y	LB_{wrap}[b]	LB_{clip}[b]
conv1	6	4	3	2	1
conv2	11	8	4	4	2
fc3	12	8	4	2	0
fc4	12	8	5	5	4

[a] Quantization errors due to rounding were ignored for this experiment.
[b] Smallest integer length where the normalized accuracy is still above 99% (see Fig. 7).

The same analysis was applied to the other benchmarks for which results are summarized in Table IV. Depending on the layer, the required accumulator range for the ACT_y constraint could potentially be reduced by 1–2 bits. However, finding these quantization solutions is non-trivial and boils down to testing more options. Finding these solutions could potentially lead to better results for even smaller accumulators bit widths.

TABLE IV
REQUIRED ACCUMULATOR RANGE (IL_{acc}) OF ALL-CNN-C (LEFT) AND ALEXNET (RIGHT) BENCHMARKS WITH DIFFERENT CONSTRAINTS.

Layer	WC_w/WC_d	ACT_w/WC_d	ACT_y	LB_{wrap}	LB_{clip}
conv1	8	6	4	1	0
conv2	13	10	6	3	2
conv3	14	11	8	5	4
conv4	16	13	9	7	5
conv5	18	15	9	8	6
conv6	18	14	8	6	5
conv7	16	14	9	7	6
conv8	16	13	9	7	6
conv9	18	14	11	10	9

Layer	WC_w/WC_d	ACT_w/WC_d	ACT_y	LB_{wrap}	LB_{clip}
conv1	11	8	6	5	5
conv2	19	13	8	8	5
conv3	18	14	8	7	5
conv4	18	14	7	6	4
conv5	17	13	7	6	4
fc6	17	13	7	6	4
fc7	16	13	7	6	4
fc8	17	13	6	6	5

VII. CONCLUSIONS

This paper presents a new quantization method for integer-based platforms without support for wide kernel accumulators. Two constraints to maximize the bit width of weights and input data for a given accumulator size are introduced. Using these constraints a layer-wise quantization heuristic for finding good fixed-point approximations is proposed. Only solutions that fully utilize the available accumulator bits are tested. We have evaluated our quantization method on three popular CNNs for image classification, and have demonstrated that 16-bit accumulators are sufficient for large CNNs. The results show that the narrow accumulators with our quantization technique can still deliver good classification performance. In future research we could exploit these findings for designing efficient accelerators and processors for NNs. Other research directions include fixed-point retraining to reduce the accuracy penalty, or finding better heuristics for faster quantization of large NNs.

REFERENCES

[1] M. Courbariaux, Y. Bengio, and J.-P. David, "Training deep neural networks with low precision multiplications," no. Section 5, pp. 1–10, 2014. [Online]. Available: http://arxiv.org/abs/1412.7024

[2] U. Köster, T. Webb, X. Wang, M. Nassar, A. K. Bansal, W. Constable, O. Elibol, S. Hall, L. Hornof, A. Khosrowshahi, C. Kloss, R. J. Pai, and N. Rao, "Flexpoint: An Adaptive Numerical Format for Efficient Training of Deep Neural Networks," *CoRR*, vol. abs/1711.02213, 2017. [Online]. Available: http://arxiv.org/abs/1711.02213

[3] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks." *CoRR*, vol. abs/1603.0, 2016. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1603.html#RastegariORF16

[4] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 818–833.

[5] P. Gysel, M. Motamedi, and S. Ghiasi, "Hardware-oriented Approximation of Convolutional Neural Networks," *CoRR*, vol. abs/1604.0, 2016. [Online]. Available: http://arxiv.org/abs/1604.03168

[6] D. Lin, S. Talathi, and S. Annapureddy, "Fixed Point Quantization of Deep Convolutional Networks," *International Conference on Machine Learning*, vol. 48, pp. 2849–2858, 2016. [Online]. Available: http://proceedings.mlr.press/v48/linb16.html

[7] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, "Angel-Eye: A Complete Design Flow for Mapping CNN onto Embedded FPGA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 1–1, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7930521/

[8] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training Deep Neural Networks with binary weights during propagations," *CoRR*, vol. abs/1511.0, 2015. [Online]. Available: http://arxiv.org/abs/1511.00363

[9] S. Anwar, K. Hwang, and W. Sung, "Fixed point optimization of deep convolutional neural networks for object recognition," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1131–1135, 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7178146/

[10] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[11] E. B. Martín Abadi, Ashish Agarwal, Paul Barham, A. D. Zhifeng Chen, Craig Citro, Greg S. Corrado, I. G. Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Y. J. Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, M. S. Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, J. S. Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, P. T. Benoit Steiner, Ilya Sutskever, Kunal Talwar, F. V. Vincent Vanhoucke, Vijay Vasudevan, M. W. Oriol Vinyals, Pete Warden, Martin Wattenberg, Yuan Yu, and X. Zheng., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: https://www.tensorflow.org/

[12] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A Framework for Empirical Study of Resource-Efficient Inference in Convolutional Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–6, 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8318896/

[13] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang, "Going Deeper with Embedded FPGA Platform for Convolutional Neural Network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '16. New York, NY, USA: ACM, 2016, pp. 26–35. [Online]. Available: http://doi.acm.org/10.1145/2847263.2847265

[14] L. Shan, M. Zhang, L. Deng, and G. Gong, "A dynamic multi-precision fixed-point data quantization strategy for convolutional neural network," in *Communications in Computer and Information Science*, vol. 666 CCIS, 2016, pp. 102–111.

[15] S. Kim and W. Sung, "Fixed-Point Simulation Utility for C and C++ Based Digital Signal Processing Programs," *IEEE Asilomar Conf. on Signals, Systems and Computers*, vol. 1, no. 11, pp. 162–166, 1994.

[16] L. Lai, N. Suda, and V. Chandra, "Deep Convolutional Neural Network Inference with Floating-point Weights and Fixed-point Activations," *CoRR*, vol. abs/1703.0, 2017. [Online]. Available: http://arxiv.org/abs/1703.03073

[17] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," 2014. [Online]. Available: http://arxiv.org/abs/1404.5997

[18] M. Courbariaux, Y. Bengio, and J.-P. David, "Low precision arithmetic for deep learning," *CoRR*, vol. abs/1412.7, 2014. [Online]. Available: http://arxiv.org/abs/1412.7024

[19] R. Yates, "Practical Considerations in Fixed-Point FIR Filter Implementations," p. 15, 2010.

[20] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep Learning with Limited Numerical Precision," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 1737–1746. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045303

[21] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for Simplicity: The All Convolutional Net," *CoRR*, vol. abs/1412.6806, 2014. [Online]. Available: http://arxiv.org/abs/1412.6806